Imperative Programs as Proofs via Game Semantics

Churchill, Martin

Award date:
2011

Awarding institution:
University of Bath

Imperative Programs as Proofs via Game Semantics

submitted by

Martin David Churchill

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Computer Science

October 2011

Abstract

Game semantics extends the Curry-Howard isomorphism to a three-way correspondence: proofs, programs, strategies. But the universe of strategies goes beyond intuitionistic logics and lambda calculus, to capture stateful programs. In this thesis we describe a logical counterpart to this extension, in which proofs denote such strategies.

The system is expressive: it contains all of the connectives of Intuitionistic Linear Logic, and first-order quantification. Use of a novel *sequoid* operator allows proofs with imperative behaviour to be expressed. Thus, we can embed first-order Intuitionistic Linear Logic into this system, Polarized Linear Logic, and an expressive imperative total programming language. We can use the first-order structure to express properties on the imperative programs.

The proof system has a tight connection with a simple game model, where games are forests of plays. Formulas are modelled as games, and proofs as history-sensitive winning strategies. We provide a strong *full and faithful completeness* result with respect to this model: each finitary strategy is the denotation of a unique analytic (cut-free) proof. Infinite strategies correspond to analytic proofs that are infinitely deep. Thus, we can normalise proofs, via the semantics.

The proof system makes novel use of the fact that the sequoid operator allows the exponential modality of linear logic to be expressed as a final coalgebra.

The work in this thesis has been presented in two conference papers, with my supervisors:

- Martin Churchill and James Laird, *A logic of sequentiality.* In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic*, volume 6247 of *Lecture Notes in Computer Science*, pages 215–229. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-15205-4-19. Based on the results in Chapter 2.

- Martin Churchill, James Laird, and Guy McCusker, *Imperative programs as proofs via game semantics.* In *Logic in Computer Science (LICS), 2011 26th Annual IEEE Symposium on*, pages 65 –74, June 2011. Based on the results in Chapters 3 and 4.

Contents

List of Figures

Acknowledgements

I am indebted to my supervisors Jim Laird and Guy McCusker for their invaluable support and wisdom throughout my PhD studies and the construction of this thesis.

I would like to thank Pierre-Louis Curien and Alessio Guglielmi for taking the time to examine my PhD thesis and for the discussions and improvements suggested during my *viva* examination.

I'd also like to thank the following individuals from the Mathematical Foundations group at Bath for providing such an enriching research environment, and lively and enlightening discussions: Martin Brain, Paola Bruscoli, Ana Calderon, Pierre Clairambault, Anupam Das, James Davenport, Alessio Guglielmi, Tom Gunderson, John Power and Cai Wingfield.

Further afield, I'd like to thank the following people for interesting conversations and inspiration: Thorsten Altenkirch, Dan Ghica, Martin Hyland, Paul Levy, John Longley, Andrzej Murawski, Luke Ong and Uday Reddy.

I would also like to express my gratitude to Makoto Takeyama, Yoshiki Kinoshita and other members of the Centre for Verification and Semantics at AIST, Japan, for hosting my wonderful visits and collaboration that culminated in the material in Appendix A.

I'd like to thank the anonymous reviewers of the two conference papers [16,17] based on the work of this thesis.

Finally, I'd like to gratefully acknowledge the financial support of the Engineering and Physical Sciences Research Council.

Chapter 1

Introduction

In which we introduce the work of this thesis.

1.1 Motivation

The Curry-Howard isomorphism [34] is a powerful theoretical and practical principle for specifying and reasoning about programs. It notes a striking correspondence between proofs and programs, of a certain kind. In the best known presentation, the proofs are those of intuitionistic logic, which correspond to typed functional programs. For example, the product operation on *types* corresponds to conjunction on *formulas* — in both cases to *pairing*. Thus, a program of type $A \times B$ is precisely a program of type A together with a program of type B; and a proof of $A \wedge B$ is precisely a proof of A together with a proof of B. Similarly, implication corresponds to the function type constructor: *modus ponens*, which concludes B from A and $A \Rightarrow B$, corresponds to function application; the cut rule to composition. This correspondence can be extended to other type constructors, noting a deep and precise connection between such proofs and programs.

The Curry-Howard correspondence can be extended to a third axis: denotational semantics. Denotational semantics provides a meaning of a program/proof as a mathematical object, e.g. a function, a relation, or (as we will see) a strategy. Fundamentally, it is *compositional* in nature: the meaning of a program/proof is defined using the meaning of its components. For example, if we wish to interpret proofs as sets and functions, we might set $[\![A \wedge B]\!]$ to be the set theoretic product of A and B; and $[\![A \Rightarrow B]\!]$ the set of functions from A to B. Thus we translate *syntactic* operations into corresponding *semantic* ones. Category theory provides an abstraction of the structure required to

give a semantic model of such logics and languages: for example, Cartesian Closed Categories correspond to models of the implication-conjunction fragment of intuitionistic logic, and the simply typed lambda calculus.

One way of giving denotational semantics to proofs and programs uses *dialogue games*. In this setting, proofs and programs are modelled as interaction sequences — dialogues between the environment and the term itself. One can construct Cartesian Closed Categories of such games, and hence model the features described above. But the categories contain richer structure. To start with, since one can control how often moves are played, games can be used to model logics with finer control over resource usage, such as linear logic [28]. Since the games models are *intensional* they can keep track of how many times an argument to a function is interrogated.

Further, the flexibility of such games models has admitted a number of *full completeness results* which state that each inhabitant of the denotational semantics is the interpretation of a proof/program [6]. Thus, the Curry-Howard correspondence extends fully to the semantics also. We might ask: how can we make this correspondence an isomorphism? A problem is that the mapping from proofs/programs to semantics is not injective. For example, a proof/program might involve a cut/composition that could be replaced by an appropriate substitution while preserving semantic meaning. Thus for the isomorphism to truly hold, we will need somehow to identify a set of "normalised" programs/proofs such that any two distinct elements are semantically distinct. This has been called *full and faithful completeness* [6]. On the proof side, clearly our normal forms must be cut-free, but this is not enough as there is often redundancy in other areas, e.g. the order in which rules are applied. The technique of *focusing* [12] enforces further discipline on proofs to help remove such redundancies.

One can also consider programs outside the purely functional setting. Examples include programs with mutable reference cells, concurrency and control effects such as exceptions and continuations. We might ask how the above correspondence between proofs, programs and semantics extends to such a setting. We can immediately address one of these axes: game semantics has been successful in providing fully complete denotational semantics for a range of such programming languages [5, 7, 9, 10, 32, 35, 44, 51].

We may also enquire into the nature of the proofs-programs correspondence in such a setting. Griffin noticed that the control operator `callcc` can be typed as Pierce's law $((A \to B) \to A) \to A$: a formula that can be added to intuitionistic logic to yield classical logic [30]. A computational calculus corresponding to classical logic was developed in [67] based on this observation.

We can address the proofs-programs correspondence by using the programs-semantics

correspondence. We can model both intuitionistic proofs and stateful programs in the same semantic framework. In particular, we can consider a model where games are simple *trees* and operations on types/formulas correspond to combinatorial operations on these trees. Programs/proofs are then represented as *strategies* for the two-player game determined by the tree. Definability results [57] ensure that each finite strategy corresponds to a program, but there are strategies (and hence programs) that do not correspond to intuitionistic proofs. In [14], Blass identified a medial rule which is not provable in Intuitionistic Linear Logic, but has an evident history-sensitive strategy.

We take the view that proof systems, as syntactic objects, should fundamentally reflect some natural semantic notion. Game semantics provides a natural interpretation of the connectives of linear logic. In this thesis, we consider a particularly simple but expressive games model, take this as a primitive, and seek the logic that best represents its structure — including a proof of Blass's medial rule, not provable in Intuitionistic Linear Logic.

One line of attack is to extend Intuitionistic Linear Logic in such a way that all (history-sensitive) strategies are definable. But we wish to do this in a harmonious manner, ideally obtaining something like the full and faithful completeness result mentioned above. If we can obtain a close correspondence between strategies and proofs, then we can embed programs in the proof system by matching their strategy semantics. Further, such a system will be of interest purely from the proofs-strategies perspective: the games model we study is a simple and natural one, exhibits rich algebraic structure and has been well studied [13, 22, 36, 51, 52, 58].

In this thesis we introduce a logic of sequentiality, where proofs denote history-sensitive strategies. This can be motivated from two angles, related as in the above discussion:

- To provide a proof system where the computational content of a proof is a stateful program
- To provide a proof system with a close semantic connection to a very simple games model — full and faithful completeness.

The logic achieves this by using Laird's sequoid operator $\oslash$ [45] as a principle logical connective. As will be seen, this connective and its dual will be enough to introduce the imperative stateful behaviour: the other connectives are standard.

The proof system we obtain is expressive. Intuitionistic Linear Logic can be embedded within it. The logic is not strictly linear, but *affine* — arguments without an exponential can be used at most once. Our justification for this is that the simple games model naturally models affine logic, and we take this semantics as a starting point.

The logic also contains first-order quantification, atoms and equality. The computational content of a proof is stateful, and we can embed an expressive total imperative programming language into the logic. We can use this together with the first-order structure to specify properties on these programs. The logic also contains a cut-free subsystem in which proofs are in some sense focused, and we will prove a full and faithful completeness result with respect to the game semantics.

1.2 Background

1.2.1 Foundations

Games for Proofs

The notion of viewing a proof as a strategy goes back a long way. The basic metaphor relates each proposition to a game between *Verifier* and *Refuter* — the job of Verifier is to show that the proposition is true, and Refuter's aim is to show that it is false. Thus, if Verifier can win every possible play, this corresponds to a proof: since each possible attempt to refute it fails. On the other hand, if there is a single way for Refuter to win, then there is a flaw in Verifier's argument, and the proposition is not proved. Thus, a proof corresponds to a *winning strategy* for Verifier — a recipe that shows him how to play, guaranteeing that he will win, however Refuter tries to outwit him.

This notion first appeared as Plato's Socratic Dialogues [64], and later in the medieval theory of *obligationes* [73]. It first appeared in its modern form in the work of Lorenzen [60], who made the metaphor precise by showing how the connectives of logic relate to corresponding constructions on games. For example:

- In the game $A \wedge B$, Refuter can choose to play in either A or B. Play then proceeds as in the chosen game. Thus, to give a winning strategy on the game $A \wedge B$, Verifier has to be able to cope with either of Refuter's choices — he must have a winning strategy for A *together with* a winning strategy for B.

- In the game $A \vee B$, Verifier can choose to play in either A or B. Play then proceeds as in the chosen game. Thus, to give a winning strategy on the game $A \vee B$, Verifier has to have a winning strategy for one or the other, at his choice.

- In the game $A \to B$, Verifier plays as in B with access to an additional resource A. That is, at a later date Verifier can chose to initiate a new game of A where Refuter plays the rôle of A-verifier, and Verifier plays the rôle of A-refuter.

- In the game $\neg A$, Verifier must try to refute A, while Refuter must try to verify it.

- In the game $\forall x.A(x)$, Refuter must chose a value v for x, and play proceeds as in $A(v)$. Thus, to give a winning strategy for $\forall x.A(x)$, Verifier must be able to cope with any choice that Refuter might make — he must have a winning strategy for $A(v)$ for each possible v.

In some of these cases it is already clear how these constructions correspond to proof rules in logical systems — e.g. the first is an interpretation of the rule

$$\frac{\vdash A \qquad \vdash B}{\vdash A \wedge B}$$

where $\vdash X$ represents a winning strategy for the game X.

The precise nature of the rules of the game and connections to formal proof systems — in particular linear logic — was made by Blass in [14]. A distinction between the multiplicative conjunction $\otimes$ and additive conjunction & can be made by varying whether Refuter is allowed to switch between the components ($\otimes$), or just choose once and for all at the start (&). Given strategies on $A \to B$ and $B \to C$ one can compose them — but in Blass's formulation this composition is not associative, and so this does not yield a category. The source of this problem is that in a particular game it may be the case that both Refuter and Verifier have potential starting moves. One solution to this is to introduce polarities, with positive and negative formulas corresponding to positive and negative games, in which all starting moves exclusively belong to Verifier or Refuter respectively [53].

As noted by Blass, there is a further problem with this model. The strategies of Blass's model are history-sensitive — the protagonists are allowed to use all of the memory at their disposal; the next move prescribed by a strategy is a function of the entire history of play so far. Resultantly, there are strategies that are not the denotation of any proof. For example, there is a history-sensitive strategy on each of the following formulas:

$$[(A \otimes B) \parr (C \otimes D)] \multimap [(A \parr C) \otimes (B \parr D)]$$
$$[A \otimes (C \& D)] \& [B \otimes (C \& D)] \& [(A \& B) \otimes C] \& [(A \& B) \otimes D] \multimap (A \& B) \otimes (C \& D)$$

These formulas are not provable in Intuitionistic Linear Logic. Thus, the interpretation is not *fully complete*.

In [6], it was shown that the natural games model for multiplicative linear logic is fully complete in this sense, if strategies make their decision based on only the last move — if they are *history-free*. Since the game metaphor is a compelling semantics of proofs, one might ask: can we enhance the underlying logics to be able to capture the full unbounded memory of the protagonists?

Games for Programs

Games and strategies also yield fruitful models of programs. Given a program type T we can consider its game interpretation $[\![T]\!]$. In this game, Verifier plays the rôle of the type T, and Refuter the rôle of the environment. Thus, if T is the type `nat -> nat`, a typical play in the corresponding game might be

- Refuter (Environment) asks for the output value
- Verifier (System) asks for the input value
- Refuter (Environment) gives the value 5 as the input
- Verifier (System) gives the value 7 as the output

Thus, a *strategy* on this game represents instructions for the System protagonist, regarding which moves it should play given the Environment moves so far. This corresponds to a program of this type. For example, the above sequence could be a play in the strategy representing the term $\lambda x.x + 2$ or $\lambda x.7$.

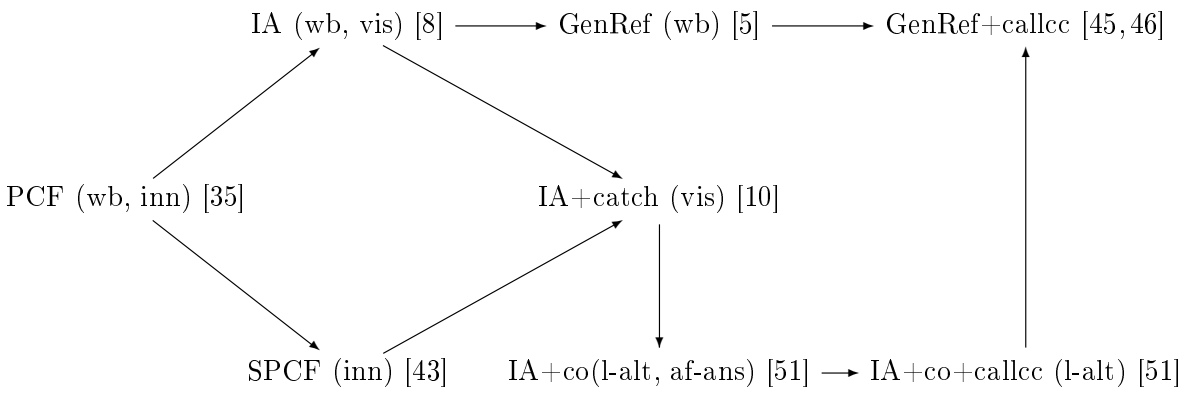
For unification of terminology, we will now refer to the Refuter-Environment player as *Opponent* and the Verifier-System player as *Player*.

An early use of this approach is the *sequential algorithms* model [13], later shown to be equivalent to a games presentation [52] (as described in [22]). In this setting, a fully abstract model of SPCF was given — this is a functional programming language over the type of naturals with a catch control operator. Thus, we already leave the range of the Curry-Howard isomorphism for intuitionistic logics: games can be used to model more expressive structure than that found in purely functional languages.

1.2.2 The Intensional Hierarchy

In the early nineties, four parties [7, 35, 65, 66] provided fully abstract (i.e. optimally precise) denotational models for the pure functional programming language PCF [69]. Three of these approaches used game semantics to do so. In the case of Hyland-Ong game semantics [35], the games were played out over an underlying *arena* consisting of a bipartite graph of Player- and Opponent- moves, together with further data such as *question-answer labelling*. Rather than the arenas representing the games themselves, they instead represented an *enabling relation* which can be used to construct the games. In particular, both protagonists would alternately play moves, provided the enabler of that move had been played sometime in the past by the other protagonist. The strategies must satisfy certain constraints, such as *innocence* (the next move prescribed

Figure 1-1: Programming Languages and Hyland-Ong Games Models



Key: Language (strategy restrictions) [reference].
Arrow denotes language embedding / weakening of strategy restrictions.

by a strategy must depend only on a certain subsequence of the play so far) and *well-bracketing* (moves played must satisfy a well-bracketing discipline with respect to the question-answer labelling).

Moreover, the tools used to construct this model would prove to be flexible indeed: by successively relaxing constraints on the games model in a natural way, languages with more expressive operations can be modelled. An early example was that of ground state, providing a model of Idealized Algol [8]. To obtain this model, it is sufficient to simply remove the innocence condition on strategies, but keeping a weaker one, called *visibility*. A ground cell can then be represented as the product of its `read` and `write` methods. This work was particularly notable because this led to the first decidability results on fragments of Idealized Algol [26], which has since led to the fruitful subfield of algorithmic game semantics [4].

Another example was given in [43], which shown that by dropping the well-bracketing condition a fully abstract model of SPCF is obtained: PCF together with a catch operator. By dropping both well-bracketing and innocence but keeping visibility, we obtain a model of Idealized Algol with local exceptions [10]. By dropping visibility as well, but keeping two weaker conditions *local alternation* and *affine-answering*, we obtain a model of a language with *coroutines* [51]. Dropping local alternation corresponds to adding higher-order references [5]; and dropping affine answering corresponds to adding full first-class continuations at all types [43,45,46]. Thus, the games models map out a rich semantic landscape of control effects and state, as seen in Figure 1-1.

Other developments in game semantics include considering nondeterministic strategies for concurrency [32] and using nominal set theory for modelling names and fresh-

ness, allowing fully abstract models of considerable subsets of ML [63].

One point in the above space is of particular interest to us. Recall that in the diagram above as we proceed from left to right we are *weakening* the constraints on the strategies, and the arena machinery was introduced precisely to make those constraints expressible. Thus, on the right hand side of this diagram, we find that the arena machinery is not necessary at all: for example, the games we find in the IA+co setting correspond to Curien-Lamarche games, or sequential algorithms [51]; and the games found in the GenRef+co to Conway games [19]. These models are strikingly simple — games are again just trees of a certain form, combined compositionally. Despite this, they yield a lot of algebraic structure, enough to model complex control and state operations. They will be our principle object of study.

1.2.3 The Curien-Lamarche Games Model

We chose to take this notion of game as our primitive one, for the following reasons:

- It is strikingly simple. A finite game is just a forest of possible plays, and the connectives on games correspond to compositional operations on these forests. Even Opponent-Player labelling can be deduced by the level a node occurs in the forest.

  This is in contrast to the (more liberal) Conway setting [21], where each node is labelled with a Player/Opponent owner. Here, even though strategies must contain alternating plays there are also non-alternating plays which are only observable when the game is placed in some larger context.

- It has rich mathematical structure. Intuitionistic Linear Logic can be modelled [14], as well as first-order state and coroutines [51].

- A strategy in any of the games models to the left of IA+co in the diagram above can be embedded inside our games model by means of forgetting the extra structure.

- Arena game semantics can in fact be modelled *inside* this model using a permissive backtracking exponential [31]. In particular, a space of innocent strategies can be constructed using an appropriate Kleisli construction. Using a different non-repetitive backtracking exponential, the fully abstract sequential algorithms model of SPCF can also be expressed in this setting [52].

- There are other equivalent representations of this formalisation of game, including locally Boolean domains [47], concrete data structures [13, 22] and graph games

[37].

- For these reasons, this model has been identified as worthy of study by Longley, who has designed an object-oriented programming language based upon it [59]. Although full general references cannot be modelled using CL-games, a restricted form can, which can be captured by a natural syntactic notion of *argument-safety* [75].

We will seek a logic where formulas correspond to CL-games and proofs of a given formula to history-sensitive strategies on the appropriate game. We wish the correspondence to be as tight as possible: in particular, we will seek full and faithful completeness.

1.2.4 The Sequoidal Operator

A categorical, axiomatic model of a language with higher-order state was given in [45]. The crux of the structure is a new *sequoid* operator $\oslash$, the algebraic properties of which reflect the sequential nature of game semantics. The games model in [5] is an instance of this categorical model. The sequoid operation has since been used to give categorical models to other languages in the intensional hierarchy, such as IA+co for which Curien-Lamarche games provide an instance [51].

Concretely, a play in $M \oslash N$ is a play in $M \otimes N$ that starts in M (Opponent may later switch between M and N). The operator has pleasant algebraic properties: to start with, it provides an *action* of the monoidal category of games $\mathcal{C}$ on its strict subcategory $\mathcal{C}_s$, and distributes over the product. Moreover, it allows a decomposition of the tensor operation, with $M \otimes N = (M \oslash N) \times (N \oslash M)$. There is an adjunction $\mathcal{C}_s(A, B \multimap C) \cong \mathcal{C}_s(A \oslash B, C)$.

In the setting of Conway games, further properties hold including an adjunction $\mathcal{C}_s(A \multimap B, C) \cong \mathcal{C}_s(B, C \oslash A)$ [45]. It turns out that such properties are sufficient to identify fully abstract models of a language with general references.

In the setting of Curien-Lamarche games, different properties hold: in particular *linear functional extensionality* [51]. Once again, this additional property is sufficient to identify fully abstract models of the coroutines language. Thus, just as constraints on the underlying arena models allow us to capture expressivity of games languages, so can the algebraic properties of the sequoid operator. This line of enquiry has continued in recent work [50].

The sequoidal operator and its algebraic properties are principal to the logic we will present here. It is the only non-standard connective in the logic, but using it we can represent proof rules with history-sensitive behaviour, and thus unlock access to the entire set of strategies available in our games model. The connective will be principal in

the sense that the interpretation of the structural connective comma will be the sequoid or its dual, and thus the meaning of all other proof rules will be tightly connected to it.

1.3 Our Contribution

1.3.1 Contributions to the Study of Proofs

We construct a first-order intuitionistic logic in which proofs have stateful computational content and correspond closely to history-sensitive strategies in the Curien-Lamarche game model. This close correspondence is made precise via *full and faithful completeness results* — each finitary strategy is the denotation of a unique analytic (cut-free) proof. Infinite strategies can be modelled using non-analytic rules, or infinitary analytic proofs.

The stateful behaviour comes from the fact that the strategies are history sensitive. Further light can be shed on this by giving an embedding of a total imperative programming language inside the proof system. We thus have a unified system where we can express both first-order logic and stateful programs, and we will be able to exploit this fact by using the first-order structure to express properties of these programs.

There are two protagonists in the underlying games model, Player and Opponent. In the logic, this will be reflected by two classes of formulas — positive and negative, corresponding to the starting protagonist in the corresponding games. This is a different notion of polarity to that found in focused proof systems [12], but we will discuss the relationship both in general and by giving a translation from Polarized Linear Logic [53].

We can normalise proofs to their analytic cut-free form via the semantics. This includes elimination of cuts, but also other admissible rules, including structural symmetry rules, weakening, and the multiplicative tensor rule (our primary rule for tensor will be different, exploiting the sequoidal structure).

The proof system is somewhat non-standard in presentation and flavour. This is a direct consequence of the fact that it models a sequential, step-by-step process, rather than more abstract notions such as information flow. The main proof-theoretic decisions when designing the system regard the choice of structural connectives and geometry of the sequents, together with the choice of proof rules themselves. Here such choices have been made to facilitate the full completeness results herein obtained with respect to the sequential, concrete games model; as well as the representation of programs and behavioural properties upon them. Thus its proof theoretic beauty from a syntactic standpoint may be lacking: future interaction with proof theorists may seek to address such concerns.

1.3.2 Contributions to Game Semantics

As well as providing a logical syntax for representing history-sensitive strategies, our work also contributes to the study of game semantics in general, in three particular areas.

Uniform history-sensitive strategies

First, we formalise the notion of *uniform history-sensitive strategy* over families of games indexed by first-order structures. The notion of a uniform family of strategies that behave independently with regards to the underlying games was used in [6] to give a games model for multiplicative linear logic, where the corresponding proofs are *history-free* (the choice of move made by the strategy depends only on the preceding Opponent-move). For example, the copycat strategy on $\alpha \multimap \alpha$ is uniform with respect to α. In a history-free setting the definition is straightforward, since at each point the only move Player can see is the previous one, which may be a concrete game, or a move from an atomic game. In a history-sensitive setting, the situation is slightly more complex as Player can have access to all previous moves, which can include an unbounded number of moves from atomic games.

Our notion of uniformity is explicitly given with respect to first-order structures. In particular, formulas are modelled as families of games indexed over first-order models, and proofs as families of strategies which are uniform with respect to the underlying model. The definition is entirely semantic, yet it yields a strong correspondence with the syntax. For example, the game $\exists x.P(x)$ at a model-valuation pair (M, v) consists of Player choosing a value $m \in M$ for x and then playing in $P(m)$. In a general family of strategies, the strategy component at a model M could pick an arbitrary element of M. Uniformity ensures that there will be some variable y such that for all models (M, v), the value $v(y)$ is picked.

We use quite a different approach to that found in other first-order games models, e.g. [55] which uses explicit first-order labels in the games.

Exponential as a Final Coalgebra

The linear exponential in our games model is history-sensitive: different occurrences of N in $!N$ can have different behaviour depending on the history of play so far — this is in contrast to the sequential algorithms sharing exponential [52]. Thus, the usual promotion rule of linear logic is not sufficient for representing all strategies. We make the observation that $!N$ is the final coalgebra of the functor $X \mapsto N \oslash X$ and express this in the logic — once again the sequoid operator plays a key rôle. There is a rule that

constructs a strategy on $A \multimap !\,B$ from a strategy on $A \multimap B \oslash A$ (its *anamorphism*). On the logic side, this follows the work of Clairambault [20] for inductive datatypes, but for the exponential operator of linear logic itself.

As well as being a mathematically clean way of representing the nature of this exponential, it is also a pleasant way to model the passing of state. To create a stream of B values $!B$ from a resource A we describe a B value for the first thread, and the resource A to be passed to the next thread: $A \multimap B \oslash A$. In particular, we can represent the strategy representing a Boolean reference cell by applying this rule to a finite strategy.

Type-theoretic presentation

In collaboration with Makoto Takeyama at AIST, Japan, we have formalised some of the work of this thesis in the proof assistant Agda [18]. This includes the foundations of game semantics in the Curien-Lamarche setting. Since these games are forests, they are well-suited to formalisation in type theory. The definitions of the multiplicatives in this setting are strikingly concise when compared to the set-theoretic counterparts, and may have pedagogical value in communicating game semantics to type theoreticians.

Another mechanisation of game semantics is Longley's Stratagem [56]. This uses continuations and universal types to construct the strategy denotation of any ML term. Our formalisation is more foundational, using dependent types in a crucial way to represent the games themselves.

1.3.3 Contribution to Programming Languages

We can view our proof system as a low-level language for describing imperative programs in a setting with expressive types. In particular, proof rules in the analytic subsystem represent move-by-move behaviour of the underlying strategies, while the proof rules in the extended system represent macro-level features such as composition, aggregating imperative objects together, and hiding part of the external interface.

The system provides a framework in which one can write imperative programs that are guaranteed to terminate, but where infinite data structures such as stacks are expressible. In particular, we will consider embeddings of total imperative languages into our system, which include expressive control features and restricted forms of higher order state. Note that our choice of games model is vital for this — if we were working in the more expressive system of Conway games, full general reference cells would be expressible, and hence cyclic reference cells, recursion, and divergence.

In particular one can express the final coalgebraic nature of $!N$ explicitly in the

programming language, in both call-by-value and call-by-name settings. In a call-by-value setting, the operator is of the following type:

$$\texttt{encaps} : \ (\texttt{s} \to \texttt{s} \times \texttt{o}) \to \texttt{s} \to (\texttt{1} \to \texttt{o})$$

One can think of `encaps f i : 1 -> o` as an object which provides an `o` value on demand, depending on its internal state. Here, `i` is the initial value of the internal state, and `f` maps the current state to an output `o` value and the new internal state. This operator has been seen before: it was the crux of encapsulating internal state in games model of an object-oriented language given in [75]. Note that `s` and `o` can be arbitrary higher-order types, so this represents a limited form of higher-order state in this total setting.

Finally, we note that we can use the first-order structure together with the imperative features. One use of this is to model programs with data-independent ground types and cells, where the only operation is equality testing.

1.4 Related Work

Several authors have previously studied proof systems inspired by semantics that are richer than the simple true/false dichotomy.

In Hintikka's independence-friendly logic [33], one can write formulas such as

$$\forall x.\exists y.\forall z.\exists w/x.P(x,y,z,w)$$

to state that y can depend on x and w on z, but w may not depend on x. This has a similar feel to the constraints discussed above on strategies, which restrict the parts of the history that is visible to Player when deciding his next move.

Japaridze's Computability Logic [39] seeks to understand computability an interactive two-player game — functions over discrete time. Unlike the games we will study, they are not alternating: Player and Opponent may play multiple moves in succession. Like our work, the simple games model is taken as a fundamental notion, and a logic is developed that accurately reflects this model. Syntactic connectives relate to semantic connectives on games, and so on. This work strictly considers strategies that are computable in nature. This model also validates Blass's examples.

Cockett, Cruttwell and Saff [21] have described a simple logic where formulas explicitly represent game trees, for Conway games. This style of presentation is closely related to our type-theoretic presentation of game semantics in Appendix A, and the cut-elimination dynamics corresponds to our type-theoretic definition of strategy composition. This in turn is related to our syntactic cut elimination procedure.

There are close links between focused proof systems [12] and game semantics. In focusing, negative and positive connectives correspond to reversible and irreversible introduction rules. Proofs proceed in negative and positive *phases* in which the reversible and irreversible rules are respectively exclusively applied. This reduces the search space of proofs, while preserving provability. The relationship to game semantics is that positive, irreversible rules correspond to Player making some move in a strategy that cannot be taken back; while the negative, reversible rules correspond to book-keeping that is not explicitly linked to a move in the corresponding strategy. To obtain our full and faithful completeness result we will need to use some similar techniques, and ours will be based on the notion of moves in games in a more explicit manner.

Girard's Ludics [29] takes focusing to its extreme, by combining all of the proof rules in a single phase into a single rule, using synthetic connectives. Thus a proof tree really does correspond to a strategy, with alternate positive and negative rules corresponding to Opponent and Player moves. Ludics has also been referred to as "untyped" in the sense that formulas themselves are abstracted away and only the relative locations matter, which correspond to the tagging in the constructions on games we will see. Our proof system does not have this property: our operations are at most binary, and there are resultantly sequences of proof rules that are syntactic book keeping that do not correspond to any move in the game model — just semantic isomorphisms.

Longley's Eriskay project [59] also uses the Curien-Lamarche games model as a starting point. This work constructs a large scale, real-world programming language with clean semantics in this simple games model. The hypothesis is that clean mathematical models lead to hygienic, consistent programming languages, as can be seen by purely functional languages. Game semantics provides clean semantic models of languages with side-effects, and the Curien-Lamarche setting is both strikingly simple and expressive. Thus, a programming language based on this model should be semantically natural, and expressive. This body of work complements ours nicely, which provides an analysis of the same model from a logical perspective.

Some models of state have used linear logic as a key tool, such as [70]. This logic also is directly inspired by a semantic model — in this case, coherence spaces. The work models *interference-free* Idealized Algol, without aliasing. Later work exploiting the sequential nature of games [9] allowed full Idealized Algol to be modelled.

We note that there has been other work in extending the Curry-Howard isomorphism to additional effects. As noted by Griffin [30], axioms for classical logic correspond to control operators. A computational calculus corresponding to classical natural deduction was given in [67]. Other extensions include using the possibility modality of modal logic to model monadic types for effects [25], and using the necessity modality to model

staged computation [68].

There has also been work in extending the Curry-Howard isomorphism to other constructs on the proof side. A notable example is the computational interpretation of full ZF classical set theory in [40] and the axiom of choice in [41]. The latter is of particular relevance, as it shows that one of the operators that can be typed by an axiom of choice is that of examining the state of an external *clock* — clearly an imperative operation.

Other approaches have been used to give games models of first-order logics, including [55]. The latter work uses explicit first-order pointers to represent information flow, which contrasts to our formalisation using lax natural transformations.

1.5 Outline of Thesis

In **Chapter 2**, we introduce the kernel of our logic, containing just the the multiplicatives, additives and units. Formulas represent finite games, and proofs total history-sensitive strategies upon them. We give a categorical model of the logic using sequoidal closed categories, of which the games model is a primary instance. We show how finitary imperative objects can be represented in the logic. We identify further axioms on the categorical model that enable a full and faithful completeness result whereby each arrow between type objects is the denotation of a unique analytic proof. Our games model satisfies these axioms. Finally, we extract from this a syntactic cut elimination procedure, which is sound with respect to any instance of the categorical model.

In **Chapter 3**, we introduce exponentials into the logic. The exponential is concretely based upon the games exponential in [36]. Since the resulting games are infinite, winning conditions must be imposed to ensure composition of totality. We observe that this exponential is the final coalgebra of $X \mapsto N \oslash X$ and use this as the key exponential rule in the logic. Using this final coalgebraic rule we model some imperative features such as a reusable Boolean reference cell. Finally, we extend the full completeness and cut elimination procedures to this setting. For infinitary strategies, the corresponding analytic proof is also infinite, which we can formalise using a final coalgebraic presentation.

In **Chapter 4**, we introduce the first-order features of our logic. In particular, we introduce atoms, quantifiers, and equality. Semantics of formulas and proofs are now *families*, indexed by the underlying first-order structure. However, the strategies must behave *uniformly* with respect to this structure, which we formalise using lax natural transformations. In the function-free setting, we can use this to show the full completeness result, where any such uniform family of winning strategies is the denotation

of a unique analytic proof. Once again, strategies that have infinite components yield infinitary analytic proofs.

In **Chapter 5**, we will show how programs and properties upon them may be expressed in our logic. We will first show how finitary call-by-name and call-by-value lambda calculi can be embedded, via Polarized Linear Logic. We add imperative constants including state and coroutines, by giving the embeddings into our logic directly. By extending the logic with an infinite-choice operator, we can express a natural number base type and all primitive recursive functions. Finally, we will show how we can use the first-order structure to represent behavioural properties upon these programs.

In **Chapter 6** we consider further directions. These include polymorphism, recursive types, other exponential structures, partiality and other game models.

In **Appendix A**, we sketch a formalisation of some of this work in the proof assistant `Agda`. In particular, we will formalise finite games, connectives on games, and strategies in type theory. We formalise the logic WS together with its semantics and full completeness procedure. We also formalise the embedding of a finitary programming language into WS, and hence its game semantics. Finally, we will provide a tool for interacting with the generated strategies.

Chapter 2

A Logic of Finite Dialogues

In this chapter we introduce the affine unit-only kernel of the Logic of Sequentiality: a sequent calculus where proofs correspond to history-sensitive strategies on finite games. We describe its categorical model and prove a strong full completeness result.

We will first describe Curien-Lamarche games and the operations on them which correspond to the syntactic connectives in our logic WS. We will then give the proof rules of WS, and equip proofs with semantics as total history-sensitive strategies. Next, we will show how finite imperative objects can be represented as proofs in WS. We will also show that multiplicative-additive Intuitionistic Linear Logic (IMALL) can be embedded inside our logic.

The formal semantics of WS will be given with respect to a categorical model: the kernel presented in this chapter can be interpreted in a *sequoidal closed category* [45] of a certain kind. The games model is an instance of this categorical model.

We will identify a set of core rules, and call proofs made up using only these rules *analytic*. In particular, analytic proofs are cut-free. We will identify further categorical axioms which identify when the model is fully and faithfully complete — i.e. each arrow between type objects is the denotation of a unique analytic proof. The Curien-Lamarche games model satisfies these axioms. Thus, this provides a normalisation procedure from proofs to analytic proofs, via the semantics. From this, a syntactic cut elimination procedure can be extracted, which is sound with respect to the categorical model.

2.1 Games and Strategies

2.1.1 Games and Strategies

Our notion of game is essentially that introduced by [14], and similar to that of [6,52]. If A is a set, let A^* denote the free monoid (set of sequences) over A, and ϵ denote the empty sequence.

Games

Definition A *game* is a tuple $(M_A, \lambda_A, b_A, P_A)$ where

- M_A is a set of moves
- $\lambda_A : M_A \to \{O, P\}$
  - We call m an *O-move* if $\lambda_A(m) = O$ and a *P-move* if $\lambda_A(m) = P$.
- $b_A \in \{O, P\}$ specifies a starting player
  - We call $s \in M_A^*$ *alternating* if s starts with a b_A-move and alternates between O-moves and P-moves. Write $M_A^{\circledast}$ for the set of such sequences.
- $P_A \subseteq M_A^{\circledast}$ is a nonempty prefix-closed set of valid plays.

Example The game of natural numbers is

$$\mathbf{N} = (q \cup \mathbb{N}, \{q \mapsto O, n \mapsto P\}, O, \{\epsilon, q\} \cup \{qn : n \in \mathbb{N}\}).$$

A maximal play consists of an Opponent move q ('which natural number are you?') followed by a Player response n for some $n \in \mathbb{N}$.

We will call a game A *negative* if $b_A = O$ and *positive* if $b_A = P$. We write $A, B, C, \ldots$ for arbitrary games; $L, M, N, \ldots$ for arbitrary negative games and $P, Q, R, \ldots$ for arbitrary positive games. Define $\neg : \{O, P\} \to \{O, P\}$ by $\neg(O) = P$ and $\neg(P) = O$.

Remark Games could be equivalently presented as a forest together with a polarity: the polarity determines whether the root moves are Opponent moves or Player moves, and alternation determines the owner of subsequent moves.

Definition A game A is *bounded* if there is some $n \in \mathbb{N}$ such that all plays in P_A have length at most n. A game A is *finite* if P_A is finite.

Strategies

As usual we define the notion of strategy as a set of traces.

Definition A *strategy* σ for a game $(M_A, \lambda_A, b_A, P_A)$ is a subset of P_A satisfying:

- If $sa \in \sigma$, then $\lambda_A(a) = P$
- If $sab \in \sigma$, then $s \in \sigma$
- If $sa, sb \in \sigma$, then $a = b$
- If $\sigma = \varnothing$ then $b_A = P$, and if $\epsilon \in \sigma$ then $b_A = O$.

We write $\mathsf{depth}(\sigma)$ for the length of the longest play in σ.

Definition A strategy on a game A is *total* if it is nonempty and whenever $s \in \sigma$ and $so \in P_A$, there is some $p \in M_A$ such that $sop \in \sigma$.

2.1.2 Connectives

We next describe various operations on games. These connectives preserve boundedness and finiteness, and will correspond to connectives in our logic. If X and Y are sets, let $X + Y = \{\mathsf{in}_1(x) : x \in X\} \cup \{\mathsf{in}_2(y) : y \in Y\}$. We use standard notation $[f, g]$ for copairing. If $s \in (X+Y)^*$ then $s|_i$ is the subsequence of s consisting of elements of the form $\mathsf{in}_i(z)$. If $X_1 \subseteq X^*$ and $Y_1 \subseteq Y^*$ let $X_1 \| Y_1 = \{s \in (X+Y)^* : s|_1 \in X_1 \wedge s|_2 \in Y_1\}$.

Empty Game

We define a negative game with no moves

$$\mathbf{1} = (\emptyset, \emptyset, O, \{\epsilon\}).$$

There is one strategy on $\mathbf{1}$ given by $\{\epsilon\}$, and this strategy is total.

We can similarly construct an empty positive game

$$\mathbf{0} = (\emptyset, \emptyset, P, \{\epsilon\}).$$

There is one strategy on $\mathbf{0}$ given by $\emptyset$ and this strategy is not total (intuitively, it is Player's turn to play first but he has no moves to play).

One-move Game

We write $\bot$ for the negative game with a maximal play of just one move:

$$\bot = (\{q\}, \{q \mapsto O\}, O, \{\epsilon, q\}\}).$$

There is a single strategy $\{\epsilon\}$ on $\bot$, and this strategy is not total.

We write $\top$ for the positive game with just one move:

$$\top = (\{q\}, \{q \mapsto P\}, P, \{\epsilon, q\}\}).$$

There is a total strategy on $\top$, given by $\{q\}$.

Product

If we consider games as forests, the product of two games is given by placing the two forests side-by-side. If $X_1 \subseteq X^*$ and $Y_1 \subseteq Y^*$ let $X_1 +^* Y_1 = \{s \in X_1 \| Y_1 : s|_1 = \epsilon \vee s|_2 = \epsilon\}$.

If N and L are negative games, define

$$N \& L = (M_N + M_L, [\lambda_N, \lambda_L], O, P_N +^* P_L).$$

Thus, on Opponent's first move he chooses to play either in N or L, and thereafter play remains in that component. A (total) strategy on $N \& L$ corresponds to a pairing of a (total) strategy on N with a (total) strategy on L.

If Q and R are positive games, define

$$Q \oplus R = (M_Q + M_R, [\lambda_Q, \lambda_R], P, P_Q +^* P_R).$$

On Player's first move he chooses to play either in Q or R, and thereafter play remains in that component. A total strategy on $Q \oplus R$ corresponds to either a total strategy on Q or a total strategy on R. The set of strategies on $Q \oplus R$ is the coalesced sum of the strategies on Q and the strategies on R, identifying the strategy $\emptyset$ in each.

Tensor

The multiplicative operators $\otimes$, $\parr$ are also played over the disjoint sum of moves.

If N and L are negative games, a play in $N \otimes L$ is an *interleaving* of a play in N together with a play in L: Opponent begins in either component, and thereafter may

switch between them. Define

$$N \otimes L = (M_N + M_L, [\lambda_N, \lambda_L], O, (P_N \| P_L) \cap M^{\circledast}_{N \otimes L}).$$

The fact that the play restricted to each component must be alternating, and that the play overall must be alternating, ensures that only Opponent may switch between components. A strategy on $N \otimes L$ does not correspond just to a pair of strategies on N and L: since strategies are history-sensitive, the choice of a move in one component can depend on play that has previously occurred in the other component.

Similarly, if Q and R are positive games, the game $Q \parr R$ consists of an interleaving where Player may switch between the two components. Define

$$Q \parr R = (M_Q + M_R, [\lambda_Q, \lambda_R], P, (P_Q \| P_R) \cap M^{\circledast}_{Q \parr R}).$$

Sequoid

The sequoid connective was introduced in [45] and its properties can be used to model stateful effects [45, 51]. Here we describe its action on Curien-Lamarche games.

A play in $N \oslash L$ is a play in $N \otimes L$ where Opponent is restricted to playing his first move in N. Similarly, a play in $Q \triangleleft R$ is a play in $Q \parr R$ that must begin in Q. We can also consider the negative game $N \triangleleft Q$, where play begins in N and Player may switch to Q after the first move and then switch between components. Finally we can consider the positive game $Q \oslash N$ where play begins in Q and it is Opponent that can switch.

If $X_1 \subseteq X^*$ and $Y_1 \subseteq Y^*$ let $X_1 \|_L Y_1 = \{s \in X_1 \| Y_1 : s|_1 = \epsilon \Rightarrow s|_2 = \epsilon\}$. Let A and B be games of arbitrary polarity. Define

$$A \between B = (M_A + M_B, [\lambda_A, \lambda_B], b_A, (P_A \|_L P_B) \cap M^{\circledast}_{A \between B}).$$

The polarity of B determines whether Player or Opponent may switch between the two components. To emphasise this, we will write $A \between B$ as $A \triangleleft B$ if B is positive (when Player may switch) and $A \oslash B$ if B is negative (when Opponent may switch).

Lifts

We can use the sequoid to add a single move to the front of a game. If N is a negative game, a play in the positive game

$$\downarrow N = \top \oslash N$$

consists of a play in N prefixed by an extra P-move. A total strategy on $\downarrow N$ corresponds to a total strategy on N. A strategy on $\downarrow N$ is either $\emptyset$ or corresponds to a strategy on N.

If P is a positive game, a play in the negative game

$$\uparrow P = \bot \lhd P$$

consists of a play in P prefixed by an extra O-move. A (total) strategy on $\uparrow P$ corresponds to a (total) strategy on P.

Negation

If A is a negative (resp. positive) game, then we can define the negation of A as a positive (resp. negative) game. This preserves the structure of the game but inverts the role of Player and Opponent. If A is a game, define

$$A^\bot = (M_A, \neg \circ \lambda_A, \neg b_A, P_A).$$

In forest notation, negation merely switches polarity.

The negation operator is involutive. There is a duality between our various operators: $\mathbf{1}^\bot = \mathbf{0}$, $\bot^\bot = \top$, $(M \otimes N)^\bot = M^\bot \parr N^\bot$, $(A \oslash N)^\bot = A^\bot \lhd N^\bot$, $(A \lhd P)^\bot = A^\bot \oslash P^\bot$, $(M \& N)^\bot = M^\bot \oplus N^\bot$.

Implication

If M and N are negative games, we can derive implication

$$M \multimap N = N \lhd M^\bot.$$

A play in $M \multimap N$ consists of a play in N interleaved with a play in a version of M where the rôles of Player and Opponent are swapped (an 'input version' of M). For example, for each function $f : \mathbb{N} \to \mathbb{N}$ we have a total strategy σ_f on $\mathbf{N} \multimap \mathbf{N}$ with maximal plays of the form

$$\mathsf{in}_2(q)\mathsf{in}_1(q)\mathsf{in}_1(n)\mathsf{in}_2(f(n))$$

for each $n \in \mathbb{N}$. Following standard notation, we will sometimes write plays in the following format:

$\mathbf{N}$	$\multimap$	$\mathbf{N}$	
		q	O
q			P
n			O
		$f(n)$	P

The horizontal positioning indicates the component (tag) of the moves, with the play as a sequence of moves running vertically.

Note that this operation represents *affine* implication: there is only one (inverted) copy of A in $A \multimap B$. Thus, there is no strategy in general on $A \multimap A \otimes A$. In the next chapter we will see how a function space can be defined where the argument can be reused.

2.1.3 Some Isomorphisms

Given two games A and B, we say that A and B are *forest isomorphic* if $b_A = b_B$ and P_A and P_B are isomorphic as forests. We will later construct categories of games, and forest isomorphisms will correspond to isomorphisms in the usual sense. Some forest isomorphisms between games are given in Figure 2-1. The (total) strategies on any two isomorphic games correspond to each other, and we will use this in the semantics of the logic WS.

Figure 2-1: Some Isomorphisms of Games

$M \otimes N \cong N \otimes M$	$P \parr Q \cong Q \parr P$
$M \otimes (N \otimes L) \cong (M \otimes N) \otimes L$	$P \parr (Q \parr R) \cong (P \parr Q) \parr L$
$M \otimes \mathbf{1} \cong M \cong M \& \mathbf{1}$	$P \parr \mathbf{0} \cong P \cong P \oplus \mathbf{0}$
$M \& N \cong N \& M$	$P \oplus Q \cong Q \oplus P$
$M \& (N \& L) \cong (M \& N) \& L$	$P \oplus (Q \oplus R) \cong (P \oplus Q) \oplus R$
$(M \otimes N) \multimap L \cong M \multimap (N \multimap L)$	$P \oslash (M \otimes N) \cong (P \oslash M) \oslash N$
$M \multimap (N \& L) \cong (M \multimap N) \& (M \multimap L)$	$(P \oplus Q) \oslash N \cong (P \oslash N) \oplus (Q \oslash N)$
$M \multimap \mathbf{1} \cong \mathbf{1} \oslash M \cong \mathbf{1}$	$\mathbf{0} \oslash M \cong \mathbf{0} \triangleleft M^{\perp} \cong \mathbf{0}$
$M \otimes N \cong (M \oslash N) \& (N \oslash M)$	$P \parr Q \cong (P \triangleleft Q) \oplus (Q \triangleleft P)$
$(M \& N) \oslash L \cong (M \oslash L) \& (N \oslash L)$	$(P \oplus Q) \triangleleft R \cong (P \triangleleft R) \oplus (Q \triangleleft R)$
$M \oslash (N \otimes L) \cong (M \oslash N) \oslash L$	$P \triangleleft (Q \parr R) \cong (P \triangleleft Q) \triangleleft R$
$M \oslash \mathbf{1} \cong M$	$P \triangleleft \mathbf{0} \cong P$
$(M \multimap N) \multimap \perp \cong (N \multimap \perp) \oslash M$	$\top \oslash (M \triangleleft Q) \cong (\top \oslash M) \triangleleft Q$
$\perp \oslash M \cong \perp$	$\top \triangleleft P \cong \top$

2.1.4 Imperative Objects as Strategies

Semantics of a full object-oriented language can be given by interpreting types as games and programs as strategies [75]. As an example, we describe the interpretation of a imperative object as a strategy on an appropriate game. We will later see how this object can be represented as a proof in our system.

For brevity, we shall abuse notation writing q_i rather than $\mathsf{in}_i(q)$.

We shall consider a simple counter object with two methods: a `void press()` method and a `nat read()` method, returning the number of times the `press` method has previously been invoked. For simplicity here, we will allow the `read` method to be called only once, and thus its type may be represented by the game $\mathbf{N}$. The type of `press` — a command that may be repeated indefinitely — may be represented as a negative game Σ^*. In this game, Opponent and Player alternately play q and a respectively. This game is defined by

$$\Sigma^* = (\{q, a\}, \{q \mapsto \mathsf{O}, a \mapsto \mathsf{P}\}, P_{\Sigma^*})$$

where P_{Σ^*} is the set of all (finite) prefixes of the infinite string $(qa)^\omega$. To combine these into an object, we use the tensor $\otimes$.

The strategy $\mathsf{count} : \Sigma^* \otimes \mathbf{N}$ representing this counter is $\{s \in P_{\Sigma^* \otimes \mathbf{N}} : \beta(s)\}$ where $\beta(s)$ holds if $s = \epsilon$ or $s = tq_1a_1$ for $\beta(t)$ or $s = tq_2m_2$ where s contains m occurrences of a_1. An example play in count is

Σ^*	$\otimes$	$\mathbf{N}$	
q			O
a			P
q			O
a			P
		q	O
		2	P

In contrast with the *history-free* strategies which denote proofs of linear logic in the model of [6], this strategy is *history-sensitive* — the move prescribed by the strategy depends on the entire play so far. It is this property which allows the state of the object to be described implicitly, as in e.g. [8].

2.2 The Logic WS

2.2.1 Proof system

We will now describe a proof system in which formulas represent (finite) games, and each proof of a formula represents a total strategy on the corresponding game.

Formulas

There are two classes of WS formulas: positive and negative. The positive and negative formulas will represent positive and negative games respectively, we speak of the *polarity* of a formula. We might read negative formulas as representing resources that must be interrogated by the environment, and positive formulas as representing systems that provide information when they choose to.

The positive and negative formulas are defined as follows:

$$\begin{array}{rcccccccccc} P & := & \mathbf{0} & | & \bot & | & P \parr Q & | & P \oplus Q & | & P \lhd Q & | & P \oslash N \\ N & := & \mathbf{1} & | & \top & | & N \otimes M & | & M \& N & | & N \oslash M & | & N \lhd P \end{array}$$

Define an operation $-^\bot$ on formulas (inverting polarity) as follows:

$$\begin{array}{rclrclrcl} \mathbf{0}^\bot & = & \mathbf{1} & (P \parr Q)^\bot & = & P^\bot \otimes Q^\bot & (P \lhd Q)^\bot & = & P^\bot \oslash Q^\bot \\ \mathbf{1}^\bot & = & \mathbf{0} & (M \otimes N)^\bot & = & M^\bot \parr N^\bot & (M \oslash N)^\bot & = & M^\bot \lhd N^\bot \\ \top^\bot & = & \bot & (P \oplus Q)^\bot & = & P^\bot \& Q^\bot & (P \oslash M)^\bot & = & P^\bot \lhd M^\bot \\ \bot^\bot & = & \top & (M \& N)^\bot & = & M^\bot \oplus N^\bot & (M \lhd P)^\bot & = & M^\bot \oslash P^\bot \end{array}$$

We define $M \multimap N = N \lhd M^\bot$, $\uparrow P = \bot \lhd P$, $\downarrow N = \top \oslash N$. We can interpret each positive (resp. negative) formula of WS as a positive (resp. negative) finite game following the constructions in Section 2.1.2.

Proof Rules

Proofs in WS of a given formula will be interpreted as total strategies on the appropriate game. A *sequent* of WS is a non-empty sequence of formulas $\vdash A_1, \ldots, A_n$. Semantically, the comma A, B will represent a left merge — $A \lhd B$ if B is positive or $A \oslash B$ if B is negative — and is therefore left-associative. For example, if M, N are negative formulas and P, Q positive formulas, the sequent

$$\vdash M, P, Q, N$$

is semantically equivalent to

$$\vdash ((M \lhd P) \lhd Q) \oslash N.$$

Thus, in the game interpretation of a sequent Γ the first move must occur in the first (or *head*) formula of Γ.

The proof rules for WS are defined in Figure 2-2. Here M, N range over negative formulas, P, Q over positive formulas, Γ, Δ over lists of formulas, Γ^* over non-empty lists of formulas and Γ^+, Δ^+ over lists of positive formulas.

The rules are partitioned into *core rules* and *admissible rules*. We say that a proof is *analytic* if it uses only the core rules. We will show that any proof is denotationally equivalent to an analytic proof. Thus, the core rules have a higher status as a choice of primitives: we could conceivably add further non-core rules (equipped with a sound semantic interpretation) if it were convenient to do so.

We make some observations regarding the core rules. First, we see that they are all additive. This is particularly striking in the case of the tensor introduction rule $\mathsf{P}_{\otimes}$. In this case the rule (additively) decomposes the plays in $M \otimes N$ into two possibilities: those that start in M and those that start in N. Thus we are explicitly modelling the fact that $\otimes$ represents an interleaving, rather than just an arbitrary monoidal structure. Second, we note that the core rules are of a specific shape: for each connective, there is a rule introducing that connective in the head position; and there are rules for eliminating certain connectives in the second position, if the head formula is $\bot$ or $\top$. The only connectives corresponding to a choice of introduction rule are $\parr$ and $\oplus$. Thus, proof search in the analytic subsystem of WS is particularly simple.

We can use the non-core rules to embed multiplicative-additive Intuitionistic Linear Logic inside WS in a compositional manner. The non-core rules have also been chosen to facilitate representing finitary imperative objects. For example, $\mathsf{P}_{\mathsf{mul}}$ to aggregating objects, and $\mathsf{P}^{-}_{\mathsf{wk}}$ to hiding part of an object's interface from the outside world. The $\mathsf{P}_{\mathsf{cut}}$ rule corresponds to composition of functions, and has been generalised appropriately based on the semantic interpretation of the structural comma connective. One might wish to generalise further, e.g. allowing negative formulas in Δ, but this is not "semantically sound". However, we may allow Γ to be empty only if Γ_1 is also empty, and this case is dealt with in the rule $\mathsf{P}^{0}_{\mathsf{cut}}$.

Focusing and Polarities

We make a brief note on polarities and reversibility, and a comparison with focused proof systems. In such systems, polarisation is used to differentiate between connectives whose

Figure 2-2: Proof rules for WS

Core rules:

$$\mathsf{P}_{\mathbf{1}}\ \frac{}{\vdash \mathbf{1}, \Gamma} \qquad \mathsf{P}_{\oslash}\ \frac{\vdash A, N, \Gamma}{\vdash A \oslash N, \Gamma} \qquad \mathsf{P}_{\triangleleft}\ \frac{\vdash A, P, \Gamma}{\vdash A \triangleleft P, \Gamma}$$

$$\mathsf{P}_{\otimes}\ \frac{\vdash M, N, \Gamma \qquad \vdash N, M, \Gamma}{\vdash M \otimes N, \Gamma} \qquad \mathsf{P}_{\parr 1}\ \frac{\vdash P, Q, \Gamma}{\vdash P \parr Q, \Gamma} \qquad \mathsf{P}_{\parr 2}\ \frac{\vdash Q, P, \Gamma}{\vdash P \parr Q, \Gamma}$$

$$\mathsf{P}_{\&}\ \frac{\vdash M, \Gamma \qquad \vdash N, \Gamma}{\vdash M \& N, \Gamma} \qquad \mathsf{P}_{\oplus 1}\ \frac{\vdash P, \Gamma}{\vdash P \oplus Q, \Gamma} \qquad \mathsf{P}_{\oplus 2}\ \frac{\vdash Q, \Gamma}{\vdash P \oplus Q, \Gamma}$$

$$\mathsf{P}^{-}_{\bot}\ \frac{\vdash \bot, \Gamma}{\vdash \bot, N, \Gamma} \qquad \mathsf{P}^{+}_{\bot}\ \frac{\vdash P}{\vdash \bot, P} \qquad \mathsf{P}^{\parr}_{\bot}\ \frac{\vdash \bot, P \parr Q, \Gamma}{\vdash \bot, P, Q, \Gamma}$$

$$\mathsf{P}^{\oslash}_{\bot}\ \frac{\vdash \bot, P \oslash N, \Gamma}{\vdash \bot, P, N, \Gamma} \qquad \mathsf{P}_{\top}\ \frac{}{\vdash \top} \qquad \mathsf{P}^{-}_{\top}\ \frac{\vdash N}{\vdash \top, N}$$

$$\mathsf{P}^{+}_{\top}\ \frac{\vdash \top, \Gamma}{\vdash \top, P, \Gamma} \qquad \mathsf{P}^{\otimes}_{\top}\ \frac{\vdash \top, M \otimes N, \Gamma}{\vdash \top, M, N, \Gamma} \qquad \mathsf{P}^{\triangleleft}_{\top}\ \frac{\vdash \top, N \triangleleft P, \Gamma}{\vdash \top, N, P, \Gamma}$$

Admissible rules:

$$\mathsf{P}^{\mathsf{T}}_{\mathbf{1}}\ \frac{\vdash \Gamma^*, \Delta}{\vdash \Gamma^*, \mathbf{1}, \Delta} \qquad \mathsf{P}^{\mathsf{T}}_{\otimes}\ \frac{\vdash \Gamma^*, M, N, \Delta}{\vdash \Gamma^*, M \otimes N, \Delta} \qquad \mathsf{P}^{-}_{\mathsf{sym}}\ \frac{\vdash \Gamma^*, M, N, \Delta}{\vdash \Gamma^*, N, M, \Delta} \qquad \mathsf{P}^{-}_{\mathsf{wk}}\ \frac{\vdash \Gamma^*, M, \Delta}{\vdash \Gamma^*, \Delta}$$

$$\mathsf{P}^{\mathsf{T}}_{\mathbf{0}}\ \frac{\vdash \Gamma^*, \mathbf{0}, \Delta}{\vdash \Gamma^*, \Delta} \qquad \mathsf{P}^{\mathsf{T}}_{\parr}\ \frac{\vdash \Gamma^*, P, Q, \Delta}{\vdash \Gamma^*, P \parr Q, \Delta} \qquad \mathsf{P}^{+}_{\mathsf{sym}}\ \frac{\vdash \Gamma^*, P, Q, \Delta}{\vdash \Gamma^*, Q, P, \Delta} \qquad \mathsf{P}^{+}_{\mathsf{wk}}\ \frac{\vdash \Gamma^*, \Delta}{\vdash \Gamma^*, P, \Delta}$$

$$\mathsf{P}_{\mathsf{mul}}\ \frac{\vdash M, \Gamma, \Delta^+ \qquad \vdash N, \Delta^+_1}{\vdash M, \Gamma, N, \Delta^+, \Delta^+_1} \qquad \mathsf{P}_{\mathsf{id}}\ \frac{}{\vdash N, N^{\bot}}$$

$$\mathsf{P}_{\mathsf{cut}}\ \frac{\vdash \Gamma^*, N^{\bot}, \Gamma_1 \qquad \vdash N, \Delta^+}{\vdash \Gamma^*, \Delta^+, \Gamma_1} \qquad \mathsf{P}^{0}_{\mathsf{cut}}\ \frac{\vdash N^{\bot} \qquad \vdash N, Q}{\vdash Q}$$

$$\mathsf{P}_{\mathsf{id}\oslash}\ \frac{\vdash N, Q, \Delta^+}{\vdash M, N, M^{\bot} \triangleleft Q, \Delta^+} \qquad \mathsf{P}^{\mathsf{T}}_{\oplus i}\ \frac{\vdash \Gamma, P_i, \Delta}{\vdash \Gamma, P_1 \oplus P_2, \Delta}$$

$$\mathsf{P}_{\multimap}\ \frac{\vdash M, \Gamma, P \qquad \vdash N, \Delta^+}{\vdash M, \Gamma, P \oslash N, \Delta^+} \qquad \mathsf{P}^{\mathsf{T}}_{\&i}\ \frac{\vdash \Gamma, M_1 \& M_2, \Delta}{\vdash \Gamma, M_i, \Delta}$$

corresponding rules are *reversible* or *irreversible* [12]. Irreversible rules act on positive formulas. An irreversible rule is one where (reading upwards) in applying the rule one must make some definite choice, a choice which could determine whether the proof search succeeds or not. Thus, additive disjunction introduction is always an irreversible rule, and in linear logic so is the tensor introduction rule, since a choice must be made regarding how the context is split (see Figure 2-3).

As we have noted, in WS the core introduction rule for tensor is additive, not multiplicative. Thus, this rule is reversible, and $\otimes$ is a negative connective. In contrast, $\parr$ is a positive connective as there are two different core introduction rules, which are not reversible. Thus, as well as the semantic motivation, we can view our distinction between positive and negative formulas in the same light as the polarities of focused systems. From a semantic viewpoint, reversible rules are those that perform book-keeping, rearranging formulas or splitting additively; while the irreversible rules actually commit to performing some specific move in a given situation.

Like focused systems, proof search in WS follows a two-phase discipline in which rules of one kind or another are exclusively applied. But the nature of these phases differ. In focused systems, the proof search alternates between negative and positive phases, in which reversible and irreversible rules are exclusively applied respectively. Analytic proof search in WS follows a different two-phase discipline, whereupon we first *decompose* the first formula of a sequent into a unit using the core introduction rules (some reversible, some irreversible), and then *collate* the tail formulas together using the core elimination rules (all of which are reversible). We will give an embedding of a particular focussed system — *Polarized Linear Logic* [53] — inside WS in Sections 2.6 and 3.4.4.

2.2.2 Interpretation of Proofs

Here we informally describe the interpretation of a proof of $\vdash \Gamma$ as a strategy on the interpretation of Γ. We will give formal categorical semantics of proofs in the next section.

- The interpretation of $\mathsf{P}_{\mathbf{1}}$ is the unique total strategy on the game $\mathbf{1}, \Gamma$ (where it is Opponent's turn to start, but there are no moves for him to play since the first move must take place in the empty game $\mathbf{1}$).

- The interpretation of $\mathsf{P}_{\top}$ is the unique total strategy on the game $\top$, where Player plays a move and the game is over.

- The interpretation of unary rules $\mathsf{P}_{\oslash}$, $\mathsf{P}_{\lhd}$, $\mathsf{P}^{\parr}_{\bot}$, $\mathsf{P}^{\oslash}_{\bot}$, $\mathsf{P}^{\otimes}_{\top}$ and $\mathsf{P}^{\lhd}_{\top}$ are based on the

fact that the game interpretation of the premise and conclusion are the same, up to retagging of moves.

- For $\mathsf{P}_{\&}$ we note that given strategies $\sigma : M, \Gamma$ and $\tau : N, \Gamma$ we can construct a strategy on $M \& N, \Gamma$ which plays as σ if Opponent's first move is in M, and as τ if Opponent's first move is in N.

- Similarly, for $\mathsf{P}_{\otimes}$ we note that given strategies $\sigma : M, N, \Gamma$ and $\tau : N, M, \Gamma$ we can construct a strategy on $M \otimes N$ which plays as σ if Opponent's first move is in M, and as τ if Opponent's first move is in N. Here we are making use of the isomorphism $M \otimes N \cong (M \oslash N) \& (N \oslash M)$ — each play in $M \otimes N$ must either start in M (and thus be a play in $M \oslash N$) or in N (and thus be a play in $N \oslash M$).

- For $\mathsf{P}_{\oplus 1}$ we note that given a strategy $\sigma : P, \Gamma$ we can construct a strategy on $P \oplus Q, \Gamma$ with Player choosing to play his first move in P and thereafter playing as σ. For $\mathsf{P}_{\oplus 2}$ Player can play his first move in Q and then play as the given strategy.

- Similarly, for the $\mathsf{P}_{\parr}$ rules, we note that in a strategy on $P \parr Q, \Gamma$ Player may choose to either play his first move in P (requiring a strategy on P, Q, Γ) or in Q (requiring a strategy on Q, P, Γ).

- The interpretation of $\mathsf{P}^{+}_{\bot}$ uses the observation that total strategies on $\bot, P = \uparrow P$ are in correspondence with total strategies on P. Similarly, the interpretation of $\mathsf{P}^{-}_{\top}$ uses the observation that total strategies on $\top, N = \downarrow N$ are in correspondence with total strategies on N.

- The interpretation of $\mathsf{P}^{-}_{\bot}$ uses the fact that the play restricted to the first two components $\bot, N$ must itself be a valid play — in particular alternating between Opponent and Player. Since Opponent plays the first move in this component he cannot also play the second move, so can never play in N — we have $\bot \oslash N \cong \bot$. Thus the set of plays in $\bot, N, \Gamma$ and $\bot, \Gamma$ are the same, up to retagging. The interpretation of $\mathsf{P}^{+}_{\top}$ is similar.

- In the cases of $\mathsf{P}^{\mathsf{T}}_{\otimes}$, $\mathsf{P}^{\mathsf{T}}_{\mathbf{1}}$, $\mathsf{P}^{\mathsf{T}}_{\parr}$, $\mathsf{P}^{\mathsf{T}}_{\mathbf{0}}$, $\mathsf{P}^{+}_{\mathsf{sym}}$ and $\mathsf{P}^{-}_{\mathsf{sym}}$, the premise and conclusion are the same game, up to retagging, and the rule can be interpreted using game isomorphisms.

- In the cases of $\mathsf{P}^{\mathsf{T}}_{\&1}$, $\mathsf{P}^{\mathsf{T}}_{\&2}$, $\mathsf{P}^{-}_{\mathsf{wk}}$, a strategy on the conclusion can be obtained by forgetting part of the strategy on the premise.

- In the cases of $\mathsf{P}^{\mathsf{T}}_{\oplus 1}$, $\mathsf{P}^{\mathsf{T}}_{\oplus 2}$, $\mathsf{P}^{+}_{\mathsf{wk}}$, a strategy on the conclusion can be obtained by using the strategy on the premise and ignoring the extra moves available to Player.
- The P_{id} rule requires a strategy on $N \multimap N$: we can use a *copycat* strategy in which Player always switches component, playing the move that Opponent previously played. The $\mathsf{P}_{\mathsf{id}_\varnothing}$ rule can be interpreted by playing copycat in the M component.
- The $\mathsf{P}_{\mathsf{cut}}$ and $\mathsf{P}^{0}_{\mathsf{cut}}$ rules can be interpreted by playing the two strategies given by the premises against each other in the N component: "parallel composition plus hiding", as we will see.
- The $\mathsf{P}_{\mathsf{mul}}$ rule can be interpreted by combining the strategies given by the premises in a multiplicative manner: Opponent's moves in M, Γ are responded to in accordance with the first premise, and moves in N in accordance with the second. The $\mathsf{P}_{\multimap}$ rule can be interpreted similarly.

We can use the above to give semantics to proofs of WS as total strategies. We will later show that any total strategy on the interpretation of a sequent Γ is the interpretation of a unique analytic proof of $\vdash \Gamma$.

2.2.3 Embedding IMALL inside WS

We next show that WS contains the multiplicative-additive fragment of Intuitionistic Linear Logic. For reference, the rules of IMALL are given in Figure 2-3. We note that each formula of IMALL can be read as a negative formula of WS (with $M \multimap N = N \lhd M^{\perp}$).

Figure 2-3: Proof rules for IMALL

$$\frac{\Gamma, M, N, \Delta \vdash L}{\Gamma, M \otimes N, \Delta \vdash L} \qquad \frac{\Gamma \vdash M \qquad \Delta \vdash N}{\Gamma, \Delta \vdash M \otimes N} \qquad \frac{\Gamma, \mathbf{1}, \Delta \vdash N}{\Gamma, \Delta \vdash N}$$

$$\frac{}{\Gamma \vdash \mathbf{1}} \qquad \frac{\Gamma \vdash M \qquad N, \Delta \vdash L}{\Gamma, M \multimap N, \Delta \vdash L} \qquad \frac{\Gamma, M \vdash N}{\Gamma \vdash M \multimap N}$$

$$\frac{\Gamma, M, \Delta \vdash C}{\Gamma, M \& N, \Delta \vdash C} \qquad \frac{\Gamma, N, \Delta \vdash C}{\Gamma, M \& N, \Delta \vdash C} \qquad \frac{\Gamma \vdash M \qquad \Gamma \vdash N}{\Gamma \vdash M \& N}$$

Proposition 2.2.1 *Let p be a proof of $M_1, \ldots, M_n \vdash N$ in IMALL. Then there exists a proof $\kappa(p)$ of $\vdash N, M_1^{\perp}, \ldots, M_n^{\perp}$ in WS.*

Proof We show that for each rule of ILL there is a derivation in WS of the conclusion from the premises.

The left $\otimes$ rule just corresponds to $\mathsf{P}^{\mathsf{T}}_{\parr}$. For the right $\otimes$ rule, with $\Gamma = G_1, \ldots, G_n$ and $\Delta = D_1, \ldots, D_m$, we duplicate the proof and use $\mathsf{P}_{\mathsf{mul}}$ in the following manner:

$$\mathsf{P}_{\otimes}\dfrac{\mathsf{P}_{\mathsf{mul}}\dfrac{\vdash M, G_1, \ldots, G_n \qquad \vdash N, D_1, \ldots, D_m}{\vdash M, N, G_1, \ldots, G_n, D_1, \ldots, D_m} \qquad \mathsf{P}^{+}_{\mathsf{sym}}\dfrac{\mathsf{P}_{\mathsf{mul}}\dfrac{\vdash N, D_1, \ldots, D_m \qquad \vdash M, G_1, \ldots, G_n}{\vdash N, M, D_1, \ldots, D_m, G_1, \ldots, G_n}}{\vdots}}{\vdash M \otimes N, G_1, \ldots, G_n, D_1, \ldots, D_m}$$

The left $\mathbf{1}$ rule just corresponds to $\mathsf{P}^{\mathsf{T}}_{\mathbf{0}}$. The right $\mathbf{1}$ rule just corresponds to $\mathsf{P}_{\mathbf{1}}$. The left $\multimap$ rule can be derived as follows:

$$\mathsf{P}_{\multimap}\dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\dfrac{\vdots}{\vdash L, D_1, \ldots, D_m, N^{\perp} \oslash M, G_1, \ldots, G_n}}{\vdash L, G_1, \ldots, G_n, N^{\perp} \oslash M, D_1, \ldots, D_m} \qquad \vdash M, G_1, \ldots, G_n}{\vdash L, D_1, \ldots, D_m, N^{\perp}}$$

The right $\multimap$ rule corresponds to $\mathsf{P}_{\triangleleft}$. The left & rules correspond to the $\mathsf{P}^{\mathsf{T}}_{\oplus}$ rules. The right & rule corresponds to $\mathsf{P}_{\&}$. ∎

Define the derived rule $\mathsf{P}_{\mathsf{mul}\otimes}$ as the derivation concluding $\vdash M \otimes N, \Delta^{+}, \Delta^{+}_{1}$ from $\vdash M, \Delta^{+}$ and $\vdash N, \Delta^{+}_{1}$ as per the translation of the IMALL right-$\otimes$ rule.

The embedding above does not cover the entire range of proofs in WS: proofs in IMALL denote history-free strategies [6], while interpretations of WS proofs may have full access to their history. For example, we can consider the medial rule as identified by Blass in [14]. This formula can be expressed in IMALL as follows:

$$((A \otimes B \multimap \perp) \otimes (C \otimes D \multimap \perp) \multimap \perp) \multimap$$
$$((\mathrm{A} \multimap \perp) \otimes (C \multimap \perp) \multimap \perp) \otimes ((B \multimap \perp) \otimes (D \multimap \perp) \multimap \perp)$$

This formula is not provable in IMALL. However, there is a history-sensitive strategy on each instantiation of the above formula: if Opponent first chooses the left hand component in the output and the right hand component in the input, Player can choose to play copycat between the copies of C, and so on. This strategy is history sensitive as the second Player move depends on both the first and second Opponent move. We will later see that any history-sensitive strategy is the denotation of a proof in WS, and so instantiations of this formula are provable in WS.

Similarly,

$$[A \otimes (C\&D)]\&[B \otimes (C\&D)]\&[(A\&B) \otimes C]\&[(A\&B) \otimes D] \multimap (A\&B) \otimes (C\&D)$$

is not provable in IMALL. However, its denotation has a history-sensitive strategy: Player chooses the component on the left-hand side based on Opponent's first move on the right-hand side. In the following play we see why this strategy is history sensitive:

$$[A \otimes (C \;\&\; D)] \;\&\; [B \otimes (C \;\&\; D)] \;\&\; \ldots \multimap (A \;\&\; B) \otimes (C \;\&\; D)$$

$$a_1 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad a_1$$

$$a_2 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad a_2$$

$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad c_1$$

$$\qquad\qquad c_1$$

Note that the response to c_1 here is in the second component, while if the first four moves took place in B the response would be in the fifth component. For the same reason, the map $(M \oslash N)\&(N \oslash M) \multimap M \otimes N$ used in our semantics of WS is history sensitive; the above strategy can be expressed in WS by decomposing $\otimes$ in this way.

The proof outline in WS is given below; the omitted branches are similar.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash A, A^\perp} \quad \mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash C\&D, C^\perp \oplus D^\perp}}{\mathsf{P}_{\mathsf{mul}}\;\vdots}}{\mathsf{P}^{\mathsf{T}}_{\parr}\dfrac{\vdash A, C\&D, A^\perp, (C^\perp \oplus D^\perp)}{\vdots}}}{\mathsf{P}^{\mathsf{T}}_{\oplus 1}\dfrac{\vdots}{\vdots}}}{\mathsf{P}_{\&}\dfrac{\vdots \qquad \vdots}{\vdots}}}{\mathsf{P}_{\otimes}\dfrac{\vdots}{\vdash (A\&B)\otimes(C\&D), (A^\perp \parr (C^\perp \oplus D^\perp)) \oplus \ldots}}$$

Thus, we can prove any instance of this formula scheme for any negative WS-formulas A,B,C and D; and we note that each finite game is expressible as a WS formula. Since WS does not have propositional variables, we cannot prove the above formula in the "general case" as a formal object in WS. As a first step in this direction, we will later introduce atoms into our proof system, which will allow us to sharpen these examples by formally identifying formulas that are not provable in IMALL but are provable in our system.

2.2.4 Imperative Objects as Proofs in WS

We now show how a bounded version of our imperative counter can be represented in WS.

Let $\oplus_0\top = \top$ and $\oplus_{n+1} = \top \oplus (\oplus_n \top)$. Then $\mathbf{N}_n = \uparrow (\oplus_n \top) = \bot \lhd (\oplus_n \top)$ represents the type of numbers at most n — there are $n+1$ analytic proofs of this formula.

We will let $\Sigma = \bot \lhd \top$ denote the type of commands that can be invoked once. This formula denotes a game with two moves — an initial Opponent move ("run") and its Player response ("done").

We let $!_0 A = \mathbf{1}$ and $!_{n+1} A = A \oslash !_n A$. We can see $!_n M$ as providing us with n copies of M. Thus $!_n \Sigma$ represents a switch that can be pressed at most n times.

We may derive a proof $\mathsf{count}_n \vdash !_n \Sigma \otimes \mathbf{N}_n$ for any n by induction, representing a finitary version of the object described in Section 2.1.4. The crux of the proof is the concluding application of the $\mathsf{P}_\otimes$ rule: this partitions interactions in $!_n \Sigma \otimes \mathbf{N}_n$ into those that start in $!_n \Sigma$ (with a `press`) and those that start in $\mathbf{N}_n$ (with a `read`).

$$\mathsf{P}_\otimes \dfrac{\dfrac{\vdots\,\mathsf{count}^1_{n,n}}{\vdash !_n\Sigma, \mathbf{N}_n} \qquad \mathsf{P}_\lhd \dfrac{\mathsf{P}^\oslash_\bot \dfrac{\mathsf{P}^+_\bot \dfrac{\mathsf{P}_\oslash \dfrac{\dfrac{\vdots\,\mathsf{count}^2_{n,n}}{\vdash \oplus_n \top, !_n\Sigma}}{\vdash (\oplus_n \top) \oslash !_n \Sigma}}{\vdash \bot, (\oplus_n \top) \oslash !_n \Sigma}}{\vdash \bot, (\oplus_n \top), !_n \Sigma}}{\vdash \mathbf{N}_n, !_n\Sigma = \bot \lhd (\oplus_n \top), !_n \Sigma}}{\mathsf{count}_n \vdash !_n \Sigma \otimes \mathbf{N}_n}$$

The proof $\mathsf{count}^1_{n,m} \vdash !_n \Sigma, \mathbf{N}_m$ for $m \geqslant n$ represents an object whose second component reveals how many times the first component has been invoked. We define $\mathsf{count}^1_{n,m}$ for $m \geqslant n$ by induction on n. The base case is simple:

$$\mathsf{P}_\mathbf{1} \dfrac{}{\mathsf{count}^1_{0,m} \vdash !_0 \Sigma, \mathbf{N}_m = \mathbf{1}, \mathbf{N}_m}$$

For $\mathsf{count}^1_{n+1,m}$, we have:

$$\mathsf{P}_\oslash \dfrac{\mathsf{P}_\otimes \dfrac{\dfrac{\vdots\,\mathsf{count}^1_{n,m}}{\vdash !_n\Sigma, \mathbf{N}_m} \qquad \mathsf{P}_\lhd \dfrac{\mathsf{P}^\oslash_\bot \dfrac{\mathsf{P}^+_\bot \dfrac{\mathsf{P}_\oslash \dfrac{\dfrac{\vdots\,\mathsf{count}^2_{n,m}}{\vdash \oplus_m \top, !_n\Sigma}}{\vdash (\oplus_m \top) \oslash (!_n \Sigma)}}{\vdash \bot, (\oplus_m \top) \oslash (!_n \Sigma)}}{\vdash \bot, \oplus_m \top, !_n \Sigma}}{\vdash \bot \lhd (\oplus_m \top), !_n \Sigma}}{\dfrac{\mathsf{P}^-_\top \dfrac{\vdash !_n\Sigma \otimes \mathbf{N}_m}{\vdash \top, !_n\Sigma \otimes \mathbf{N}_m}}{\mathsf{P}^\otimes_\top \dfrac{\vdash \top, !_n\Sigma, \mathbf{N}_m}{\mathsf{P}_\oslash \dfrac{\vdash (\top \oslash !_n \Sigma), \mathbf{N}_m}{\mathsf{P}_\oslash \dfrac{\vdash (\top \oslash !_n \Sigma) \oslash \mathbf{N}_m}{\mathsf{P}^+_\bot \dfrac{\vdash \bot, (\top \oslash !_n \Sigma) \oslash \mathbf{N}_m}{\mathsf{P}^\oslash_\bot \dfrac{\vdash \bot, \top \oslash !_n \Sigma, \mathbf{N}_m}{\mathsf{P}^\oslash_\bot \dfrac{\vdash \bot, \top, !_n \Sigma, \mathbf{N}_m}{\mathsf{P}_\lhd \dfrac{\vdash \bot \lhd \top, !_n \Sigma, \mathbf{N}_m}{\mathsf{count}^1_{n+1,m} \vdash !_{n+1}\Sigma, \mathbf{N}_m = (\bot \lhd \top) \oslash !_n \Sigma, \mathbf{N}_m}}}}}}}}}$$

The proof $\mathsf{count}^2_{n,m} \vdash \oplus_m \top, !_n\Sigma$ for $m \geqslant n$ uses the difference between m and n to determine the result of `read`: it responds with the number $m-n$ in its first component.

$$\mathsf{P}_{\oplus 2}\ \dfrac{\begin{array}{c}\vdots\ \mathsf{count}^2_{n,n+a}\\ \hline \vdash \oplus_{n+a}\top, !_n\Sigma\end{array}}{\mathsf{count}^2_{n,n+a+1} \vdash \top \oplus (\oplus_{n+a}\top), !_n\Sigma} \qquad \mathsf{P}_{\oplus 1}\ \dfrac{\mathsf{P}^-_{\top}\ \dfrac{\begin{array}{c}\vdots\ !\mathsf{com}_{n+1}\\ \hline \vdash !_{n+1}\Sigma\end{array}}{\vdash \top, !_{n+1}\Sigma}}{\mathsf{count}^2_{n+1,n+1} \vdash \top \oplus (\oplus_n \top), !_{n+1}\Sigma}$$

Then $\mathsf{count}^2_{0,0}$ and $!\mathsf{com}_n$ are given as follows:

$$\mathsf{P}^-_{\top}\ \dfrac{\mathsf{P}_{\mathbf{1}}\ \dfrac{}{\vdash \mathbf{1}}}{\mathsf{count}^2_{0,0} \vdash \top, \mathbf{1}} \qquad \mathsf{P}_{\oslash}\ \dfrac{\begin{array}{c}\mathsf{P}_{\top}\ \dfrac{\begin{array}{c}\vdots\ !\mathsf{com}_n\\ \hline \vdash !_n\Sigma\end{array}}{\vdash \top, !_n\Sigma}\\ \mathsf{P}_{\oslash}\ \dfrac{}{\vdash \top \oslash !_n\Sigma}\\ \mathsf{P}^{\oslash}_{\bot}\ \dfrac{}{\vdash \bot, \top \oslash !_n\Sigma}\\ \mathsf{P}^{\bot}_{\top}\ \dfrac{}{\vdash \bot, \top, !_n\Sigma}\\ \mathsf{P}_{\lhd}\ \dfrac{}{\vdash \bot \lhd \top, !_n\Sigma}\end{array}}{!\mathsf{com}_{n+1} \vdash (\bot \lhd \top) \oslash !_n\Sigma} \qquad \mathsf{P}_{\mathbf{1}}\ \dfrac{}{!\mathsf{com}_0 \vdash \mathbf{1}}$$

The interpretation of count_n is the history-sensitive strategy on $!_n\Sigma \otimes \mathbf{N}_n$ that behaves as a counter, as in Section 2.1.4.

2.3 Categorical Semantics of WS

We now describe a categorical model of WS together with a principal example based on games and strategies. It will be based on the notion of *sequoidal closed category* [45]. We use notation $\eta : F \Rightarrow G : \mathcal{C} \to \mathcal{D}$ to mean η is a natural transformation from F to G with $F, G : \mathcal{C} \to \mathcal{D}$.

2.3.1 Categories of Games

First, we present some game categories that will be the intended instance of our categorical model. Objects in these categories will be negative games, and an arrow $A \to B$ will be a strategy on $A \multimap B$. We can compose strategies using "parallel composition plus hiding". Suppose $\sigma : A \multimap B$ and $\tau : B \multimap C$, define

$$\sigma \| \tau = \{s \in (M_A + M_B + M_C)^* : s|_1 \in P_A \wedge s|_2 \in P_B \wedge s|_3 \in P_C\}$$

and set

$$\tau \circ \sigma = \{s|_{1,3} : s \in \sigma \| \tau\}.$$

It is well-known that $\tau \circ \sigma$ is a well-formed strategy on $A \multimap C$ (see e.g. [6]).

Proposition 2.3.1 *Composition is associative, and there is an identity $A \to A$ given by the copycat strategy: $\{s \in P_{A \multimap A} : \gamma(s)\}$ where $\gamma(s)$ holds if and only if $t|_1 = t|_2$ for all even-lengthed prefixes t of s.*

Definition The category $\mathcal{G}$ has negative games as objects, and a map $\sigma : A \to B$ is a strategy on $A \multimap B$ with composition and identity as above.

This category has been studied extensively in e.g. [22, 52, 58], and has equivalent presentations using *graph games* [37] and locally Boolean domains [47].

If A, B and C are bounded, $\sigma : A \multimap B$ and $\tau : B \multimap C$ are total then $\tau \circ \sigma$ is also total. We can thus construct subcategories of total strategies.

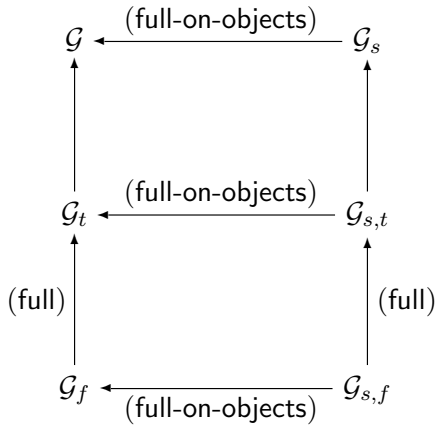
Definition The category $\mathcal{G}_t$ has bounded negative games as objects and total strategies as maps. The category $\mathcal{G}_f$ is the full subcategory of $\mathcal{G}_t$ containing only finite games.

A map $\sigma : A \to B$ is *strict* if it responds to Opponent's first move with a move in A, if it responds at all. Strict strategies are closed under composition and the identity is strict.

Definition The category $\mathcal{G}_s$ has negative games as objects and strict strategies as maps. The category $\mathcal{G}_{s,t}$ has bounded negative games as objects and strict, total strategies as maps. The category $\mathcal{G}_{s,f}$ is the full subcategory of $\mathcal{G}_{s,t}$ containing only finite games.

Isomorphisms in $\mathcal{G}$ correspond to forest isomorphisms and all isomorphisms are total and strict [54].

Each of the above categories can be endowed with a symmetric monoidal closed structure, given by $(I, \otimes, \multimap)$ where I is the empty game $\mathbf{1}$ and the actions of $\otimes$ and $\multimap$ on objects are defined as in Section 2.1.2. The following relationships hold:



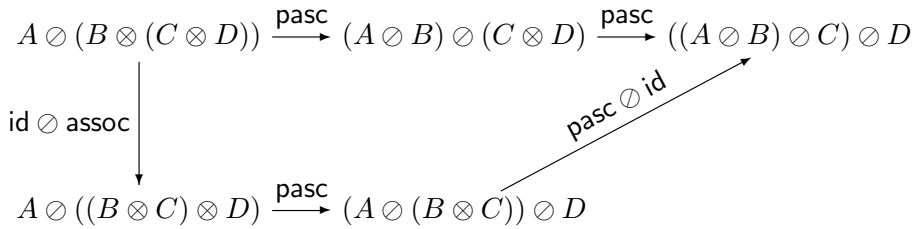
An arrow from category $\mathcal{A}$ to category $\mathcal{B}$ indicates that $\mathcal{A}$ is a symmetric monoidal closed subcategory of $\mathcal{B}$. Some full and full-on-objects subcategories are identified.

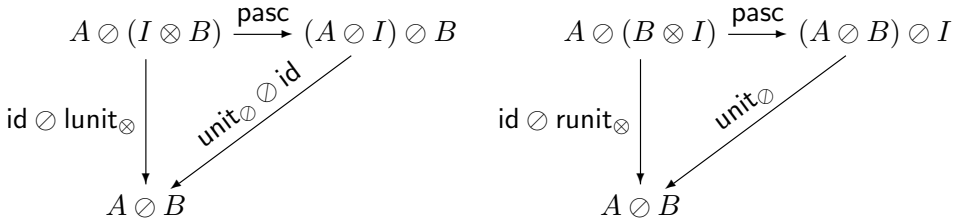
2.3.2 WS-categories

The notion of *sequoidal closed category* was first introduced in [45].

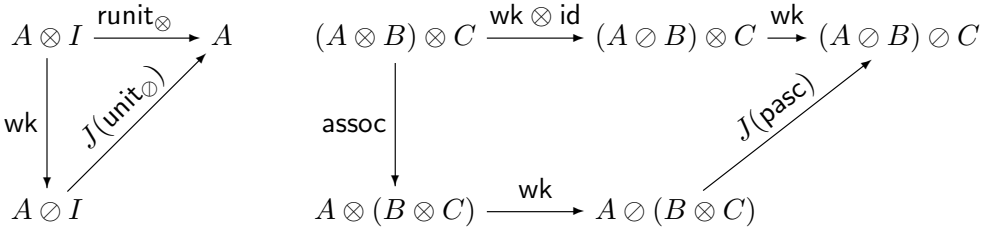
Definition A *sequoidal category* consists of:

- A symmetric monoidal category $(\mathcal{C}, I, \otimes)$ (we will call the relevant isomorphisms $\mathsf{assoc} : (A \otimes B) \otimes C \cong A \otimes (B \otimes C)$, $\mathsf{lunit}_\otimes : A \otimes I \cong A$, $\mathsf{runit}_\otimes : I \otimes A \cong I$ and $\mathsf{sym} : A \otimes B \cong B \otimes A$)
- A category $\mathcal{C}_s$
- A right-action $\oslash$ of $\mathcal{C}$ on $\mathcal{C}_s$. That is, a functor $\_ \oslash \_ : \mathcal{C}_s \times \mathcal{C} \to \mathcal{C}_s$ with natural isomorphisms $\mathsf{unit}_\oslash : A \oslash I \cong A$ and $\mathsf{pasc} : A \oslash (B \otimes C) \cong (A \oslash B) \oslash C$ satisfying the following coherence conditions [38]:





- A functor $J : \mathcal{C}_s \to \mathcal{C}$
- A natural transformation $\mathsf{wk} : J(\_) \otimes \_ \Rightarrow J(\_ \oslash \_)$ satisfying further coherence conditions [45]:



Definition An *inclusive sequoidal category* is a sequoidal category in which $\mathcal{C}_s$ is a full-on-objects subcategory of $\mathcal{C}$ containing the monoidal isomorphisms and wk; J is the inclusion functor; and J reflects isomorphisms.

We can identify this structure in our categories of games: we can extend the left-merge operator $\oslash$ to an action $\mathcal{G}_s \times \mathcal{G} \to \mathcal{G}_s$. If $\sigma : A \to B$ and $\tau : C \to D$, $\sigma \oslash \tau : A \oslash C \to B \oslash D$ plays as σ between A and B and as τ between C and D. Note that this only yields a valid strategy on $(A \oslash C) \multimap (B \oslash D)$ if σ is strict. The isomorphisms pasc and $\mathsf{unit}_{\oslash}$ exist, and there is a natural copycat strategy $\mathsf{wk} : M \otimes N \to M \oslash N$ in $\mathcal{G}_s$, all satisfying the required axioms [51]. The functor J reflects isomorphisms as the inverse of strict isomorphisms are strict. Thus $(\mathcal{G}, \mathcal{G}_s)$ forms an inclusive sequoidal category; as do $(\mathcal{G}_t, \mathcal{G}_{s,t})$ and $(\mathcal{G}_f, \mathcal{G}_{s,f})$.

Definition An inclusive sequoidal category is *Cartesian* if $\mathcal{C}_s$ has finite products preserved by J (we will write t_A for the unique map $A \to 1$). It is *decomposable* if the natural transformations $\mathsf{dec} = \langle \mathsf{wk}, \mathsf{wk} \circ \mathsf{sym} \rangle : A \otimes B \Rightarrow (A \oslash B) \times (B \oslash A) : \mathcal{C}_s \times \mathcal{C}_s \to \mathcal{C}_s$ and $\mathsf{dec}^0 = \mathsf{t}_I : I \Rightarrow 1 : \mathcal{C}_s$ are isomorphisms (so, in particular, $(\mathcal{C}, \otimes, I)$ is an affine SMC).

A Cartesian sequoidal category is *distributive* if the natural transformations $\mathsf{dist} = \langle \pi_1 \oslash \mathsf{id}_C, \pi_2 \oslash \mathsf{id}_C \rangle : (A \times B) \oslash C \Rightarrow (A \oslash C) \times (B \oslash C) : \mathcal{C}_s \times \mathcal{C}_s \times \mathcal{C} \to \mathcal{C}_s$ and $\mathsf{dist}_0 = \mathsf{t}_{1 \oslash C} : 1 \oslash C \Rightarrow 1 : \mathcal{C} \to \mathcal{C}_s$ are isomorphisms.

We write $\mathsf{dist}^0 : I \oslash C \cong I$ for the isomorphism $(\mathsf{dec}^0)^{-1} \circ \mathsf{dist}_0 \circ (\mathsf{dec}^0 \oslash \mathsf{id})$.

In the game categories defined above, $M \& N$ is a product of M and N, and the empty game I is a terminal object as well as the monoidal unit. The decomposability and distributivity isomorphisms above exist as natural copycat morphisms [51].

Definition A *sequoidal closed category* is an inclusive sequoidal category where $\mathcal{C}$ is symmetric monoidal closed and the map $f \mapsto \Lambda(f \circ \mathsf{wk})$ defines a natural isomorphism $\Lambda_s : \mathcal{C}_s(B \oslash A, C) \Rightarrow \mathcal{C}_s(B, A \multimap C)$.

We can show that $\mathcal{G}$ is sequoidal closed, with the internal hom given by $\multimap$ [51].

In any sequoidal closed category, define $\mathsf{app}_s : (A \multimap B) \oslash A \to B$ as $\Lambda_s^{-1}(\mathsf{id})$, and $\mathsf{app} : (A \multimap B) \otimes A \to B = \Lambda^{-1}(\mathsf{id})$, noting that $\mathsf{app} = \mathsf{app}_s \circ \mathsf{wk}$. If $f : A \to B$ let $\Lambda_I(f) : I \to A \multimap B$ denote the name of f, i.e. $\Lambda(f \circ \mathsf{runit}_{\otimes})$. We let Λ_I^{-1} denote the inverse operation. Recall that in any symmetric monoidal closed category the following hold:

- $\Lambda(f \circ g \circ (h \otimes j)) = (j \multimap f) \circ \Lambda(g) \circ h$
- $\Lambda_I(f \circ g) = (\mathsf{id} \multimap f) \circ \Lambda_I(g) = (g \multimap \mathsf{id}) \circ \Lambda_I(f)$
- $\Lambda_I^{-1}(f) \circ g = \Lambda_I^{-1}((g \multimap \mathsf{id}) \circ f)$, $g \circ \Lambda_I^{-1}(f) = \Lambda_I^{-1}((\mathsf{id} \multimap g) \circ f)$
- $\langle \Lambda_I f, \Lambda_I g \rangle = \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ \Lambda_I \langle f, g \rangle$

Proposition 2.3.2 *In any sequoidal closed category,* $\multimap$ *restricts to a functor* $\mathcal{C}^{\mathsf{op}} \times \mathcal{C}_s \to \mathcal{C}_s$ *with natural isomorphisms* $\mathsf{unit}_{\multimap} : I \multimap A \cong A$ *and* $\mathsf{pasc}_{\multimap} : A \otimes B \multimap C \cong A \multimap (B \multimap C)$ *in* $\mathcal{C}_s$.

Proof We need to show that if g is in $\mathcal{C}_s$ then $f \multimap g$ is in $\mathcal{C}_s$. But $f \multimap g = \Lambda(g \circ \mathsf{app} \circ (\mathsf{id} \otimes f)) = \Lambda(g \circ \mathsf{app}_s \circ \mathsf{wk} \circ (\mathsf{id} \otimes f)) = \Lambda(g \circ \mathsf{app}_s \circ (\mathsf{id} \oslash f) \circ \mathsf{wk}) = \Lambda_s(g \circ \mathsf{app}_s \circ (\mathsf{id} \oslash f))$ which is in $\mathcal{C}_s$.

In any symmetric monoidal category the isomorphisms $\mathsf{unit}_{\multimap}$ and $\mathsf{pasc}_{\multimap}$ exist, but we must show that they are strict.

- $\mathsf{unit}_{\multimap} : I \multimap A \to A$ is given by $\mathsf{app} \circ \mathsf{runit}_{\otimes}^{-1}$. This $\mathsf{app}_s \circ \mathsf{wk} \circ \mathsf{runit}_{\otimes}^{-1} = \mathsf{app}_s \circ \mathsf{unit}_{\oslash}^{-1}$ which is a map in $\mathcal{C}_s$.
- $\mathsf{pasc}_{\multimap} : A \otimes B \multimap C \cong A \multimap (B \multimap C)$ is given by $\Lambda(\Lambda(\mathsf{app} \circ \mathsf{assoc})) = \Lambda(\Lambda(\mathsf{app}_s \circ \mathsf{wk} \circ \mathsf{assoc})) = \Lambda(\Lambda(\mathsf{app}_s \circ \mathsf{pasc}^{-1} \circ \mathsf{wk} \circ (\mathsf{wk} \otimes \mathsf{id})) = \Lambda(\Lambda(\mathsf{app}_s \circ \mathsf{pasc}^{-1} \circ \mathsf{wk}) \circ \mathsf{wk}) = \Lambda_s(\Lambda_s(\mathsf{app}_s \circ \mathsf{pasc}^{-1}))$ which is in $\mathcal{C}_s$.

The inverses of the above maps are strict as J preserves isomorphisms. ∎

In distributive, decomposable sequoidal closed categories we can also define the following natural transformations:

- The isomorphism $\mathsf{psym} : (A \oslash B) \oslash C \cong (A \oslash C) \oslash B$ given by $\mathsf{pasc} \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1}$.
- The isomorphism $\mathsf{psym}_{\multimap} : C \multimap (B \multimap A) \cong B \multimap (C \multimap A)$ given by $\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}_{\multimap}^{-1}$
- The isomorphism $\mathsf{dist}_{\multimap} : A \multimap (B \times C) \to (A \multimap B) \times (A \multimap C)$ given by $\langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle$, whose inverse is $\Lambda\langle \mathsf{app} \circ (\pi_1 \otimes \mathsf{id}), \mathsf{app} \circ (\pi_2 \otimes \mathsf{id}) \rangle$. This isomorphism exists in any monoidal closed category with products.
- The map $\mathsf{af} : A \Rightarrow I$ given by $(\mathsf{dec}^0)^{-1} \circ \mathsf{t}_A$.
- The isomorphism $\mathsf{dist}^0_{\multimap} : A \multimap I \to I$ given by af whose inverse is $\Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af}))$. We must check that these are inverses: $\mathsf{af} \circ \Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af})) = \mathsf{id}$ as both are maps into the terminal object, and $\Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af})) \circ \mathsf{af} = \Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{af} \otimes \mathsf{id}) \circ (\mathsf{af} \otimes \mathsf{id})) = \Lambda(\mathsf{app}) = \mathsf{id}$ as required. We know that $\mathsf{runit}_{\otimes} \circ (\mathsf{af} \otimes \mathsf{id}) \circ (\mathsf{af} \otimes \mathsf{id}) = \mathsf{app}$ as both are maps into the terminal object.

We can use the structure described above to model the negative connectives of WS. We will represent positive connectives indirectly, inspired by the fact that strategies on

the positive game P correspond to strategies on the negative game $\uparrow P = P^{\perp} \multimap \perp$ where $\perp$ is the one-move game. The object $\perp$ satisfies a special property: an internalised version of *linear functional extensionality* [3] which shows that any map from a linear function space $A \multimap B$ into $\perp$ can be decomposed into an argument A and a map from B into $\perp$:

Definition An object $\perp$ in a sequoidal closed category satisfies *linear functional extensionality* if the natural transformation $\mathsf{lfe} : (B \multimap \perp) \oslash A \Rightarrow (A \multimap B) \multimap \perp : \mathcal{C} \times \mathcal{C}^{\mathsf{op}} \to \mathcal{C}_s$ given by $\Lambda_s(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1})$ is an isomorphism.

Related properties have been considered for games models that are not history-sensitive [3, 7], using a tensor rather than sequoid decomposition.

The linear functional extensionality property holds in the Curien-Lamarche games model [51], but fails in other sequoidal closed categories (e.g. Conway games). In a sense, it is an algebraic representation of *local alternation*: using linear functional extensionality we can give a natural isomorphism $\mathsf{abs} : \perp \oslash A \cong \perp$ by noticing that $\perp \oslash A \cong (I \multimap \perp) \oslash A \cong (A \multimap I) \multimap \perp \cong I \multimap \perp \cong \perp$, and thus setting $\mathsf{abs} = \mathsf{unit}_{\multimap} \circ ((\mathsf{dist}^0_{\multimap})^{-1} \multimap \mathsf{id}) \circ \mathsf{lfe} \circ (\mathsf{unit}^{-1}_{\circ} \oslash \mathsf{id})$. In the Conway setting, this isomorphism doesn't hold: consider a play in a strict strategy on $\perp \multimap A \oslash \perp$, in which after the first two moves Opponent may return to A.

Definition A *WS-category* is a distributive, decomposable sequoidal closed category with an object $\perp$ satisfying linear functional extensionality.

Proposition 2.3.3 *$(\mathcal{G}, \mathcal{G}_s)$, $(\mathcal{G}_t, \mathcal{G}_{s,t})$, $(\mathcal{G}_f, \mathcal{G}_{s,f})$ all enjoy the structure of a WS-category.*

The category of locally Boolean domains [47] is another example of a WS-category.

2.3.3 Semantics of Formulas and Sequents

Let $\mathcal{C}$ be a WS-category. We give semantics of both positive and negative formulas as objects in $\mathcal{C}$ below. Note that in our semantics of formulas, $[\![A]\!] = [\![A^{\perp}]\!]$. However, the polarity of a formula will affect the type of the denotation of proofs of that formula, as will be seen.

$$
\begin{array}{lcllcl}
[\![\mathbf{1}]\!] & = & I & [\![\mathbf{0}]\!] & = & I \\
[\![\bot]\!] & = & \bot & [\![\top]\!] & = & \bot \\
[\![M \otimes N]\!] & = & [\![M]\!] \otimes [\![N]\!] & [\![P \parr Q]\!] & = & [\![P]\!] \otimes [\![Q]\!] \\
[\![M \& N]\!] & = & [\![M]\!] \times [\![N]\!] & [\![P \oplus Q]\!] & = & [\![P]\!] \times [\![Q]\!] \\
[\![M \oslash N]\!] & = & [\![M]\!] \oslash [\![N]\!] & [\![P \triangleleft Q]\!] & = & [\![P]\!] \oslash [\![Q]\!] \\
[\![M \triangleleft Q]\!] & = & [\![Q]\!] \multimap [\![M]\!] & [\![P \oslash N]\!] & = & [\![N]\!] \multimap [\![P]\!]
\end{array}
$$

We consider our list-connective comma to be a binary operator associating to the left. Then $[\![A, B]\!]$ is $[\![A]\!] \oslash [\![B]\!]$ if A and B are of the same polarity, and $[\![B]\!] \multimap [\![A]\!]$ otherwise.

2.3.4 Semantics of Contexts

A context is a (possibly empty) list of formulas. If Γ is a context, we give semantics $[\![\Gamma]\!]^b$ for $b \in \{+, -\}$ as endofunctors on $\mathcal{C}_s$ below.

$$
\begin{array}{lcllcl}
[\![\epsilon]\!]^+ & = & \mathsf{id} & [\![\epsilon]\!]^- & = & \mathsf{id} \\
[\![\Gamma, M]\!]^+ & = & [\![M]\!] \multimap [\![\Gamma]\!]^+ & [\![\Gamma, P]\!]^- & = & [\![P]\!] \multimap [\![\Gamma]\!]^- \\
[\![\Gamma, P]\!]^+ & = & [\![\Gamma]\!]^+ \oslash [\![P]\!] & [\![\Gamma, M]\!]^- & = & [\![\Gamma]\!]^- \oslash [\![M]\!]
\end{array}
$$

Proposition 2.3.4 *For any sequent A, Γ we have $[\![A, \Gamma]\!] = [\![\Gamma]\!]^b([\![A]\!])$ where b is the polarity of A.*

Proof A simple induction on Γ. ∎

Proposition 2.3.5 *For any context Γ, $[\![\Gamma]\!]^b$ preserves products.*

Proof We can construct isomorphisms $\mathsf{dist}_{\mathsf{b},\Gamma} : [\![\Gamma]\!]^b(A \times B) \cong [\![\Gamma]\!]^b(A) \times [\![\Gamma]\!]^b(B)$ and $\mathsf{dist}^0_{\mathsf{b},\Gamma} : [\![\Gamma]\!]^b(I) \cong I$ by induction on Γ.

$$
\begin{array}{lcllcl}
\mathsf{dist}^0_{-,\epsilon} & = & \mathsf{id} & \mathsf{dist}^0_{+,\epsilon} & = & \mathsf{id} \\
\mathsf{dist}^0_{-,\Gamma,P} & = & \mathsf{dist}^0_{-\circ} \circ (\mathsf{id} \multimap \mathsf{dist}^0_{-,\Gamma}) & \mathsf{dist}^0_{-,\Gamma,N} & = & \mathsf{dist}^0 \circ (\mathsf{dist}^0_{-,\Gamma} \oslash \mathsf{id}) \\
\mathsf{dist}^0_{+,\Gamma,N} & = & \mathsf{dist}^0_{-\circ} \circ (\mathsf{id} \multimap \mathsf{dist}^0_{-,\Gamma}) & \mathsf{dist}^0_{+,\Gamma,P} & = & \mathsf{dist}^0 \circ (\mathsf{dist}^0_{-,\Gamma} \oslash \mathsf{id}) \\
\mathsf{dist}_{-,\epsilon} & = & \mathsf{id} & \mathsf{dist}_{+,\epsilon} & = & \mathsf{id} \\
\mathsf{dist}_{-,\Gamma,P} & = & \mathsf{dist}_{-\circ} \circ (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}) & \mathsf{dist}_{-,\Gamma,N} & = & \mathsf{dist} \circ (\mathsf{dist}_{-,\Gamma} \oslash \mathsf{id}) \\
\mathsf{dist}_{+,\Gamma,N} & = & \mathsf{dist}_{-\circ} \circ (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}) & \mathsf{dist}_{+,\Gamma,P} & = & \mathsf{dist} \circ (\mathsf{dist}_{-,\Gamma} \oslash \mathsf{id})
\end{array}
$$

We only need to show that $[\![\Gamma]\!]^b(\pi_i) \circ \mathsf{dist}^{-1}_{b,\Gamma} = \pi_i$, i.e. $[\![\Gamma]\!](\pi_i) = \pi_i \circ \mathsf{dist}_{b,\Gamma}$. Suppose $b = -$. We proceed by induction on Γ.

- If $\Gamma = \epsilon$ then $[\![\epsilon]\!](\pi_i) = \pi_i = \pi_i \circ \mathsf{id} = \pi_i \circ \mathsf{dist}_{b,\epsilon}$.

- If $\Gamma = \Gamma', N$ then $[\![\Gamma]\!](\pi_i) = [\![\Gamma']\!](\pi_i) \oslash \mathsf{id} = \pi_i \circ \mathsf{dist}_{-,\Gamma} \oslash \mathsf{id} = (\pi_i \oslash \mathsf{id}) \circ (\mathsf{dist}_{-,\Gamma'} \oslash \mathsf{id}) = \pi_i \circ \mathsf{dist} \circ (\mathsf{dist}_{-,\Gamma'} \oslash \mathsf{id}) = \pi_i \circ \mathsf{dist}_{-,\Gamma}$ as required.
- If $\Gamma = \Gamma', P$ then $[\![\Gamma]\!](\pi_i) = \mathsf{id} \multimap [\![\Gamma']\!](\pi_i) = \mathsf{id} \multimap \pi_i \circ \mathsf{dist}_{-,\Gamma} = (\mathsf{id} \multimap \pi_i) \circ (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}) = \pi_i \circ \mathsf{dist}_{-\circ} \circ (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}) = \pi_i \circ \mathsf{dist}_{-,\Gamma}$ as required.

The case for $b = +$ is entirely similar. ∎

2.3.5 Semantics of Proofs

While the semantics of formulas are independent of polarity, semantics of proofs are not. If $p \vdash A, \Gamma$ is a proof, we define $[\![p \vdash A, \Gamma]\!]$ as an arrow $\mathcal{C}(I, [\![A, \Gamma]\!])$ in the case that A is negative, and as an arrow in $\mathcal{C}([\![A, \Gamma]\!], \bot)$ in the case that A is positive. Semantics of the core rules are given in Figure 2-4 and the other rules in Figures 2-5 and 2-6.

In the semantics of $\mathsf{P}_{\mathsf{cut}}$ we use an additional construction. If $\tau : I \to [\![N, \Delta]\!]$ define (strict) $\tau^{\circ -}_{M,\Gamma} : [\![M, \Gamma, N^\bot]\!] \to [\![M, \Gamma, \Delta]\!]$ to be $\mathsf{unit}_{-\circ} \circ (\tau \multimap \mathsf{id}_{[\![M,\Gamma]\!]})$ if $|\Delta| = 0$ and $\mathsf{pasc}^n_{-\circ} \circ (\Lambda^{-n}\Lambda_I^{-1}\tau \multimap \mathsf{id}_{[\![M,\Gamma]\!]})$ if $|\Delta| = n + 1$. Define (strict) $\tau^{\circ +}_{P,\Gamma} : [\![P, \Gamma, \Delta]\!] \to [\![P, \Gamma, N^\bot]\!]$ to be $(\mathsf{id}_{[\![P,\Gamma]\!]} \oslash \tau) \circ \mathsf{unit}^{-1}_{\oslash}$ if $|\Delta| = 0$ and $(\mathsf{id} \oslash \Lambda^{-n}\Lambda_I^{-1}\tau) \circ ((\mathsf{id}_{[\![P,\Gamma]\!]} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1})^n$ if $|\Delta| = n + 1$. In some of the rules in Figure 2-6 we omit some pasc isomorphisms for clarity.

2.4 Full Completeness

We now prove a strong *full completeness* result for the games model of WS: every total strategy on a game denoted by a formula is the denotation of a unique analytic proof of that formula (i.e. one which only uses the core rules). This exhibits a strong correspondence between syntax and semantics, and establishes admissibility of all non-core rules.

We will first describe this proof-extraction procedure in our model of games and strategies, and then give categorical axioms sufficient for a model of WS to enjoy this full completeness result.

2.4.1 Reification of Strategies

We define a procedure reify which transforms a strategy on a formula object into a proof of that formula. It may be seen as a semantics-guided proof search procedure: given a strategy σ on the interpretation of Γ, reify finds a proof which denotes it. Reading upwards, the procedure first seeks to decompose the head formula into a unit (nullary connective) using the head introduction rules. If this unit is $\mathbf{1}$, we are done. It cannot

Figure 2-4: Categorical Semantics for WS (core rules)

$$\mathsf{P}_{\mathbf{1}}\ \frac{}{(\mathsf{dist}^{0}_{-,\Gamma})^{-1} : [\![\vdash \mathbf{1}, \Gamma]\!]} \qquad \mathsf{P}_{\top}\ \frac{}{\mathsf{id}_{\perp} : [\![\vdash \top]\!]}$$

$$\mathsf{P}_{\otimes}\ \frac{\sigma : [\![\vdash M, N, \Gamma]\!] \qquad \tau : [\![\vdash N, M, \Gamma]\!]}{[\![\Gamma]\!]^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle \sigma, \tau \rangle : [\![\vdash M \otimes N, \Gamma]\!]} \qquad \mathsf{P}_{\&}\ \frac{\sigma : [\![\vdash M, \Gamma]\!] \qquad \tau : [\![\vdash N, \Gamma]\!]}{\mathsf{dist}^{-1}_{-,\Gamma} \circ \langle \sigma, \tau \rangle : [\![\vdash M \& N, \Gamma]\!]}$$

$$\mathsf{P}_{\parr 2}\ \frac{\sigma : [\![\vdash Q, P, \Gamma]\!]}{\sigma \circ [\![\Gamma]\!]^{+}(\mathsf{wk} \circ \mathsf{sym}) : [\![\vdash P \parr Q, \Gamma]\!]} \qquad \mathsf{P}_{\parr 1}\ \frac{\sigma : [\![\vdash P, Q, \Gamma]\!]}{\sigma \circ [\![\Gamma]\!]^{+}(\mathsf{wk}) : [\![\vdash P \parr Q, \Gamma]\!]}$$

$$\mathsf{P}_{\oplus 1}\ \frac{\sigma : [\![\vdash P, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^{+}(\pi_1) : [\![\vdash P \oplus Q, \Delta]\!]} \qquad \mathsf{P}_{\oplus 2}\ \frac{\sigma : [\![\vdash Q, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^{+}(\pi_2) : [\![\vdash P \oplus Q, \Delta]\!]}$$

$$\mathsf{P}^{\parr}_{\perp}\ \frac{\sigma : [\![\vdash \perp, P \parr Q, \Gamma]\!]}{[\![\Gamma]\!]^{-}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id})) \circ \sigma : [\![\vdash \perp, P, Q, \Gamma]\!]} \qquad \mathsf{P}^{+}_{\perp}\ \frac{\sigma : [\![\vdash P]\!]}{\Lambda_I(\sigma) : [\![\vdash \perp, P]\!]}$$

$$\mathsf{P}^{\oslash}_{\perp}\ \frac{\sigma : [\![\vdash \perp, P \oslash N, \Gamma]\!]}{[\![\Gamma]\!]^{-}(\mathsf{lfe}^{-1}) \circ \sigma : [\![\vdash \perp, P, N, \Gamma]\!]} \qquad \mathsf{P}^{-}_{\top}\ \frac{\sigma : [\![\vdash N]\!]}{\mathsf{unit}_{\multimap} \circ (\sigma \multimap \mathsf{id}) : [\![\vdash \top, N]\!]}$$

$$\mathsf{P}^{\otimes}_{\top}\ \frac{\sigma : [\![\vdash \top, M \otimes N, \Gamma]\!]}{\sigma \circ [\![\Gamma]\!]^{+}((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_{\multimap}) : [\![\vdash \top, M, N, \Gamma]\!]} \qquad \mathsf{P}^{\triangleleft}_{\top}\ \frac{\sigma : [\![\vdash \top, N \triangleleft P, \Gamma]\!]}{\sigma \circ [\![\Gamma]\!]^{+}(\mathsf{lfe}) : [\![\vdash \top, N, P, \Gamma]\!]}$$

$$\mathsf{P}^{-}_{\perp}\ \frac{\sigma : [\![\vdash \perp, \Gamma]\!]}{[\![\Gamma]\!]^{-}(\mathsf{abs}^{-1}) \circ \sigma : [\![\vdash \perp, N, \Gamma]\!]} \qquad \mathsf{P}_{\triangleleft}\ \frac{\sigma : [\![\vdash A, P, \Gamma]\!]}{\sigma : [\![\vdash A \triangleleft P, \Gamma]\!]}$$

$$\mathsf{P}^{+}_{\top}\ \frac{\sigma : [\![\top, \Gamma]\!]}{\sigma \circ [\![\Gamma]\!]^{+}(\mathsf{abs}) : [\![\top, P, \Gamma]\!]} \qquad \mathsf{P}_{\oslash}\ \frac{\sigma : [\![\vdash A, N, \Gamma]\!]}{\sigma : [\![\vdash A \oslash N, \Gamma]\!]}$$

Figure 2-5: Categorical Semantics for WS (other rules, part 1)

$$\frac{\sigma : [\![\vdash M', \Gamma, M, N, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{psym}) \circ \sigma : [\![\vdash M', \Gamma, N, M, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash P, \Gamma, M, N, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{psym}_{\multimap}) : [\![\vdash P, \Gamma, N, M, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash M, \Gamma, P, Q, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{psym}_{\multimap}) \circ \sigma : [\![\vdash M, \Gamma, Q, P, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash P', \Gamma, P, Q, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{psym}) : [\![\vdash P', \Gamma, Q, P, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash P, \Gamma, M, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_{\multimap}^{-1}) : [\![\vdash P, \Gamma, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash N, \Gamma, M, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})) \circ \sigma : [\![\vdash N, \Gamma, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash M, \Gamma, \Delta]\!]}{[\![\Delta]\!]^-((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_{\multimap}^{-1}) \circ \sigma : [\![\vdash M, \Gamma, P, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash P, \Gamma, \Delta^+]\!]}{\sigma \circ [\![\Delta]\!]^-(\mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})) : [\![\vdash P, \Gamma, Q, \Delta^+]\!]}$$

$$\frac{\sigma : [\![\vdash N, \Gamma, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{unit}_{\oslash}^{-1}) \circ \sigma : [\![\vdash N, \Gamma, \mathbf{1}, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash P, \Gamma, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{unit}_{\multimap}) : [\![\vdash P, \Gamma, \mathbf{1}, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash M, \Gamma, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{unit}_{\multimap}^{-1}) \circ \sigma : [\![\vdash M, \Gamma, \mathbf{0}, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash P, \Gamma, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{unit}_{\oslash}) : [\![\vdash P, \Gamma, \mathbf{0}, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash M', \Gamma, M, N, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{pasc}) \circ \sigma : [\![\vdash M', \Gamma, M \otimes N, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash P, \Gamma, M, N, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{pasc}_{\multimap}) : [\![\vdash P, \Gamma, M \otimes N, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash P', \Gamma, P, Q, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{pasc}^{-1}) : [\![\vdash P', \Gamma, P \otimes Q, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash M, \Gamma, P, Q, \Delta]\!]}{[\![\Delta]\!]^+(\mathsf{pasc}_{\multimap}^{-1}) \circ \sigma : [\![\vdash M, \Gamma, P \otimes Q, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash M, \Gamma, P_i, \Delta]\!]}{[\![\Delta]\!]^-(\pi_i \multimap \mathsf{id}) \circ \sigma : [\![\vdash M, \Gamma, P_1 \oplus P_2, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash Q, \Gamma, P_i, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{id} \oslash \pi_i) : [\![\vdash Q, \Gamma, P_1 \oplus P_2, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash N, \Gamma, M_1 \& M_2, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{id} \oslash \pi_i) \circ \sigma : [\![\vdash N, \Gamma, M_i, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash Q, \Gamma, M_1 \& M_2, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\pi_i \multimap \mathsf{id}) : [\![\vdash Q, \Gamma, M_i, \Delta]\!]}$$

Figure 2-6: Categorical Semantics for WS (other rules, part 2)

$$\mathsf{P}_{\mathsf{mul}}\ \frac{\sigma : [\![\vdash M, \Gamma, \Delta^+]\!] \qquad \tau : [\![\vdash N, \Delta_1^+]\!]}{\Lambda_I\Lambda(\mathsf{wk} \circ (\Lambda_I^{-1}(\sigma) \otimes \Lambda_I^{-1}(\tau))) : [\![\vdash M, \Gamma, N, \Delta^+, \Delta_1^+]\!]}$$

$$\mathsf{P}_{\mathsf{cut}}\ \frac{\sigma : [\![\vdash M, \Gamma, N^\perp, \Gamma_1]\!] \qquad \tau : [\![\vdash N, \Delta^+]\!]}{[\![\Gamma_1]\!]^-(\tau^{\circ-}_{M,\Gamma}) \circ \sigma : [\![\vdash M, \Gamma, \Delta^+, \Gamma_1]\!]}$$

$$\mathsf{P}_{\mathsf{cut}}\ \frac{\sigma : [\![\vdash P, \Gamma, N^\perp, \Gamma_1]\!] \qquad \tau : [\![\vdash N, \Delta^+]\!]}{\sigma \circ [\![\Gamma_1]\!]^+(\tau^{\circ+}_{P,\Gamma}) : [\![\vdash P, \Gamma, \Delta^+, \Gamma_1]\!]}$$

$$\mathsf{P}_{\mathsf{id}}\ \frac{}{\Lambda_I(\mathsf{id}) : [\![\vdash N, N^\perp]\!]}$$

$$\mathsf{P}^0_{\mathsf{cut}}\ \frac{\sigma : [\![\vdash N^\perp]\!] \qquad \tau : [\![\vdash N, Q]\!]}{\sigma \circ \Lambda_I^{-1}(\tau) : [\![\vdash Q]\!]}$$

$$\mathsf{P}_{\multimap}\ \frac{\sigma : [\![\vdash M, \Gamma, P]\!] \qquad \tau : [\![\vdash N, \Delta^+]\!]}{\mathsf{psym}_{\multimap} \circ \Lambda_I(\Lambda_I^{-1}(\tau) \multimap \Lambda_I^{-1}(\sigma)) : [\![\vdash M, \Gamma, P \oslash N, \Delta^+]\!]}$$

$$\mathsf{P}_{\mathsf{id}\oslash}\ \frac{\sigma : [\![\vdash N, Q, \Delta^+]\!]}{\Lambda_I\Lambda((\mathsf{id} \oslash \Lambda^{-1}\Lambda_I^{-1}(\sigma) \circ \mathsf{sym}) \circ \mathsf{pasc}_{\oslash} \circ \mathsf{wk} \circ \mathsf{sym}) : [\![\vdash M, N, M^\perp \lhd Q, \Delta^+]\!]}$$

be $\mathbf{0}$, as there are no (total) strategies on this game. If the unit is $\top$ or $\bot$, the procedure then consolidates the tail of Γ into a single formula, using the core elimination rules. Once this is done, the head unit is removed using $\mathsf{P}^+_\bot$ or $\mathsf{P}^-_\top$, strictly decreasing the size of the sequent. These steps are then repeated until termination.

- The case $\Gamma = \mathbf{0}, \Gamma'$ is impossible: there are no total strategies on this game.
- If $\Gamma = \mathbf{1}, \Gamma'$ then σ must be the empty strategy, since it is the unique total strategy on this game. This is the interpretation of the proof $\mathsf{P}_{\mathbf{1}}$.
- If $\Gamma = \top$ then σ must similarly be the unique total strategy on this game, i.e. the interpretation of $\mathsf{P}_\top$.
- If $\Gamma = \top, P, \Gamma'$ then σ can never play in P since if it did the play restricted to $\top, P$ would not be alternating. Thus σ is a strategy on $\top, \Gamma'$. We can call reify inductively yielding a proof of $\vdash \top, \Gamma'$, and apply $\mathsf{P}^+_\top$ to yield a proof of $\top, P, \Gamma$.
- If $\Gamma = \top, N, P, \Gamma'$ then σ is a total strategy on $\top, N \lhd P, \Gamma$ up to retagging and we can proceed inductively using $\mathsf{P}^\lhd_\top$. If $\Gamma = \top, N, M, \Gamma'$ we can proceed similarly, using $\mathsf{P}^\otimes_\top$.

- If $\Gamma = \top, N$ then σ is a total strategy on $\downarrow N$: we can strip off the first move yielding a total strategy on N, apply reify inductively yielding a proof of $\vdash N$, and finally apply $\mathsf{P}^-_\top$ yielding a proof of $\vdash \top, N$.

- The case $\Gamma = \bot$ is impossible: there are no total strategies on this game. Other cases where $\bot$ is the head formula proceed as with $\top$: if the tail is a single positive formula, we remove the first move and apply $\mathsf{P}^+_\bot$, otherwise we shorten the tail using $\mathsf{P}^-_\bot$, $\mathsf{P}^\oslash_\bot$ or $\mathsf{P}^{\parr}_\bot$.

- If $\Gamma = A \oslash N, \Gamma'$ then σ is also a strategy on A, N, Γ. We can call reify inductively yielding a proof of $\vdash A, N, \Gamma$ that denotes σ, and apply $\mathsf{P}_\oslash$. We can proceed similarly in the following case $\Gamma = A \lhd P, \Gamma'$.

- If $\Gamma = M \& N, \Gamma'$ then we can split σ into those plays that start with M and those that start with N. This yields total strategies on M, Γ and N, Γ respectively, which we can reify inductively and apply $\mathsf{P}_\&$.

- If $\Gamma = M \otimes N, \Gamma'$ then we can split σ into those plays that start with M and those that start with N. This yields total strategies on M, N, Γ and N, M, Γ respectively, which we can reify inductively and apply $\mathsf{P}_\otimes$.

- If $\Gamma = P \oplus Q, \Gamma$ then σ specifies a first move that must either be in P or in Q. In the former case, we have a strategy on P, Γ and can reify inductively, finally applying $\mathsf{P}_{\oplus 1}$. In the latter case, we have a strategy on Q, Γ and can reify inductively and apply $\mathsf{P}_{\oplus 2}$. The case of $\Gamma = P \parr Q, \Gamma$ is similar.

In Proposition 2.4.3 we show that reify is well defined by giving a measure on sequents that decreases on each call to the inductive hypothesis.

2.4.2 Example of Reification

We next give an example of reification. We write reify_Γ for the operation of reification from total strategies on $[\![\Gamma]\!]$ to proofs of $\vdash \Gamma$. We will sometimes write proofs in term notation, so $\mathsf{P}_\otimes(p_1, p_2)$ denotes the proof obtained by applying the $\mathsf{P}_\otimes$ rule to the proofs p_1 and p_2.

We can restrict the counter strategy given in Section 2.1.4 to the bounded types given in Section 2.2.4. In particular, we consider the counter strategy on the denotation of $\Sigma \otimes \mathbf{N}_1 = (\bot \lhd \top) \otimes (\bot \lhd (\top \oplus \top))$. We will show how this strategy is reified to a proof of this formula in WS.

To recall, the game $[\![\Sigma]\!]$ acts as a button that can be pushed (we write q for the opening move and a for its response). The game $[\![\mathbf{N}_1]\!]$ acts as a Boolean (with an

opening move q and two responses 0 and 1). A play in $[\![\Sigma \otimes \mathbf{N}_1]\!]$ is either a play in $[\![\Sigma]\!]$ followed by a play in $[\![\mathbf{N}_1]\!]$, or *vice versa* (since the games are only two moves long, the switching behaviour simplifies to this situation). Consider the strategy $\sigma : [\![\Sigma \otimes \mathbf{N}_1]\!]$ which responds to the Boolean component with 0 if the switch has not been pressed, and 1 if it has been pressed. Then the maximal plays in σ are $q_1 a_1 q_2 1_2$ and $q_2 0_2 q_1 a_1$. Prefixes of the former form a strategy σ_1 on $[\![\Sigma, \mathbf{N}_1]\!] = [\![\Sigma \oslash \mathbf{N}_1]\!]$ and prefixes of the latter form a strategy σ_2 on $[\![\mathbf{N}_1, \Sigma]\!] = [\![\mathbf{N}_1 \oslash \Sigma]\!]$. Then

- $\mathsf{reify}_{\Sigma \otimes \mathbf{N}_1}(\sigma) = \mathsf{P}_\otimes(\mathsf{reify}_{\Sigma, \mathbf{N}_1}(\sigma_1), \mathsf{reify}_{\mathbf{N}_1, \Sigma}(\sigma_2))$ as the outermost connective of the formula is $\otimes$.
- For $\mathsf{reify}(\sigma_1)$:
  - $\mathsf{reify}_{\Sigma, \mathbf{N}_1}(\sigma_1) = \mathsf{reify}_{\bot \lhd \top, \mathbf{N}_1}(\sigma_1) = \mathsf{P}_\lhd(\mathsf{reify}_{\bot, \top, \mathbf{N}_1}(\sigma_1))$ as the outermost connective of the head is $\lhd$.
  - $\mathsf{reify}_{\bot, \top, \mathbf{N}_1}(\sigma_1) = \mathsf{P}^\oslash_\bot(\mathsf{reify}_{\bot, \top \oslash \mathbf{N}_1}(\sigma_1))$ as the head formula is $\bot$ and the tail consists of a positive formula followed by a negative formula.
  - $\mathsf{reify}_{\bot, \top \oslash \mathbf{N}_1}(\sigma_1) = \mathsf{P}^+_\bot(\mathsf{reify}_{\top \oslash \mathbf{N}_1}(\sigma_1'))$ where σ_1' is the strategy on $[\![\top \oslash \mathbf{N}_1]\!]$ whose maximal plays are given by $a_1 q_2 1_2$ — we have removed the first move from σ_1.
  - $\mathsf{reify}_{\top \oslash \mathbf{N}_1}(\sigma_1') = \mathsf{P}_\oslash(\mathsf{reify}_{\top, \mathbf{N}_1}(\sigma_1'))$ as the outermost connective is $\oslash$.
  - $\mathsf{reify}_{\top, \mathbf{N}_1}(\sigma_1') = \mathsf{P}^-_\top(\mathsf{reify}_{\mathbf{N}_1}(\sigma_1''))$ where $\sigma_1'' : [\![\mathbf{N}_1]\!]$ has maximal plays of the form $q1$ — we have removed the first move from σ_1' (and relabelled).
  - $\mathsf{reify}_{\mathbf{N}_1 = \bot \lhd (\top \oplus \top)}(\sigma_1'') = \mathsf{P}_\lhd(\mathsf{reify}_{\bot, \top \oplus \top})$ as the outermost connective is $\lhd$.
  - $\mathsf{reify}_{\bot, \top \oplus \top}(\sigma_1'') = \mathsf{P}^+_\bot(\mathsf{reify}_{\top \oplus \top}(\sigma_1'''))$ where $\sigma_1''' : [\![\top \oplus \top]\!]$ has a single maximal play 1.
  - To calculate $\mathsf{reify}_{\top \oplus \top}(\sigma_1''')$ we notice that the outer connective is $\oplus$, so we must look at the first move of σ_1''' to determine which rule to use. Since σ_1''' begins with a move in the right hand component, we note that $\mathsf{reify}_{\top \oplus \top}(\sigma_1''') = \mathsf{P}_{\oplus 2}(\mathsf{reify}_\top(a))$ where a is the unique total strategy on $[\![\top]\!]$.
  - Then $\mathsf{reify}_\top(a) = \mathsf{P}_\top$.

  So $\mathsf{reify}(\sigma_1) = \mathsf{P}_\lhd(\mathsf{P}^\oslash_\bot(\mathsf{P}^+_\bot(\mathsf{P}_\oslash(\mathsf{P}^-_\top(\mathsf{P}_\lhd(\mathsf{P}^+_\bot(\mathsf{P}_{\oplus 2}(\mathsf{P}_\top)))))))).$
- We can similarly calculate

$$\mathsf{reify}_{\mathbf{N}_1 \oslash \Sigma}(\sigma_2) = \mathsf{P}_\lhd(\mathsf{P}^\oslash_\bot(\mathsf{P}^+_\bot(\mathsf{P}_\oslash(\mathsf{P}_{\oplus 1}(\mathsf{P}^-_\top(\mathsf{P}_\lhd(\mathsf{P}^+_\bot(\mathsf{P}_\top)))))))).$$

So, $\mathsf{reify}(\sigma)$ is the following proof:

$$\dfrac{\dfrac{\mathsf{P}_{\triangleleft}\ \dfrac{\mathsf{P}_{\top}\ \dfrac{}{\vdash \top}}{\mathsf{P}_{\oplus 2}\ \dfrac{\vdash \top \oplus \top}{\mathsf{P}^{+}_{\bot}\ \dfrac{\vdash \bot, \top \oplus \top}{\vdash \mathbf{N}_1 = \bot \triangleleft (\top \oplus \top)}}}}{\mathsf{P}^{-}_{\top}\ \dfrac{\vdash \top, \mathbf{N}_1}{\mathsf{P}_{\oslash}\ \dfrac{\vdash \top \oslash \mathbf{N}_1}{\mathsf{P}^{+}_{\bot}\ \dfrac{\vdash \bot, \top \oslash \mathbf{N}_1}{\mathsf{P}^{\oslash}_{\bot}\ \dfrac{\vdash \bot, \top, \mathbf{N}_1}{\mathsf{P}_{\triangleleft}\ \dfrac{\vdash \Sigma, \mathbf{N}_1 = \bot \triangleleft \top, \mathbf{N}_1}{\mathsf{P}_{\otimes}\ \dfrac{}{\vdash \Sigma \otimes \mathbf{N}_1}}}}}}}} \qquad \dfrac{\dfrac{\mathsf{P}_{\top}\ \dfrac{}{\vdash \top}}{\mathsf{P}^{+}_{\bot}\ \dfrac{\vdash \bot, \top}{\mathsf{P}_{\triangleleft}\ \dfrac{\vdash \Sigma = \bot \triangleleft \top}{\mathsf{P}^{-}_{\top}\ \dfrac{\vdash \top, \Sigma}{\mathsf{P}_{\oplus 1}\ \dfrac{\vdash \top \oplus \top, \Sigma}{\mathsf{P}_{\oslash}\ \dfrac{\vdash (\top \oplus \top) \oslash \Sigma}{\mathsf{P}^{+}_{\bot}\ \dfrac{\vdash \bot, (\top \oplus \top) \oslash \Sigma}{\mathsf{P}^{\oslash}_{\bot}\ \dfrac{\vdash \bot, \top \oplus \top, \Sigma}{\mathsf{P}_{\triangleleft}\ \dfrac{\vdash \mathbf{N}_1, \Sigma = \bot \triangleleft (\top \oplus \top), \Sigma}{\mathsf{P}_{\otimes}\ \dfrac{}{\vdash \Sigma \otimes \mathbf{N}_1}}}}}}}}}$$

2.4.3 Complete WS-categories

In the above procedure we use properties of the game model that do not follow from the definition of a WS-category. We now give further categorical axioms in the style of [3], capturing the properties of a WS-category which enable full completeness. Rather than identifying a class of categorical models with many or varied examples (precluded by the strength of the result itself), these axioms allow us to give a rigorous and abstract proof of full completeness using the structure of a WS-category.

Definition A *complete* WS-category is a WS-category such that:

1a The unique map $i : \varnothing \Rightarrow \mathcal{C}(I, \bot)$ is a bijection.

1b The map $\mathsf{d} = [\lambda f.f \circ \pi_1, \lambda g.f \circ \pi_2] : \mathcal{C}(M, \bot) + \mathcal{C}(N, \bot) \Rightarrow \mathcal{C}(M \times N, \bot)$ is a bijection. (π-*atomicity* [3]).

2 The map $\_ \multimap \bot : \mathcal{C}(I, M) \Rightarrow \mathcal{C}(M \multimap \bot, I \multimap \bot)$ is a bijection.

These axioms capture the properties of determinacy, totality and the object $\bot$.

Proposition 2.4.1 *$\mathcal{G}_t$ is a complete WS-category.*

Proof Axiom ($1a$) holds as there are no total strategies on the game $\bot$. Axiom ($1b$) holds since total strategies $M \times N \to \bot$ correspond to total strategies on the positive game $M^{\bot} \oplus N^{\bot}$ and any such strategy must either play its first move in $M^{\bot}$ or $N^{\bot}$ (but not both). Axiom (2) holds since maps $M \multimap \bot \to I \multimap \bot$ correspond to total strategies on the positive game $\downarrow M$ which correspond to total strategies on M. ∎

$\mathcal{G}_f$ is also a complete WS-category, but $\mathcal{G}$ is not: for example, partiality allows the empty strategy on $I \multimap \bot$ violating axiom (1a).

Figure 2-7: Reification of Strategies as Analytic Proofs

$$
\begin{array}{lcl}
\mathsf{reify}_{\mathbf{1},\Gamma}(\sigma) & = & \mathsf{P}_{\mathbf{1}} \\
\mathsf{reify}_{\bot,N,\Gamma}(\sigma) & = & \mathsf{P}^{-}_{\bot}(\mathsf{reify}_{\bot,\Gamma}([\![\Gamma]\!]^{-}(\mathsf{abs}) \circ \sigma)) \\
\mathsf{reify}_{\bot,P}(\sigma) & = & \mathsf{P}^{+}_{\bot}(\mathsf{reify}_{P}(\Lambda_I^{-1}(\sigma))) \\
\mathsf{reify}_{\bot,P,Q,\Gamma}(\sigma) & = & \mathsf{P}^{\parr}_{\bot}(\mathsf{reify}_{\bot,P \parr Q,\Gamma}([\![\Gamma]\!]^{-}((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_{\multimap} \circ \sigma))) \\
\mathsf{reify}_{\bot,P,N,\Gamma}(\sigma) & = & \mathsf{P}^{-}_{\bot}(\mathsf{reify}_{\bot,P \oslash N,\Gamma}([\![\Gamma]\!]^{-}(\mathsf{lfe}) \circ \sigma)) \\
\mathsf{reify}_{M \& N,\Gamma}(\sigma) & = & \mathsf{P}_{\&}(\mathsf{reify}_{M,\Gamma}(\pi_1 \circ \mathsf{dist}_{-,\Gamma} \circ \sigma))(\mathsf{reify}_{N,\Gamma}((\pi_2 \circ \mathsf{dist}_{-,\Gamma} \circ \sigma)) \\
\mathsf{reify}_{M \otimes N,\Gamma}(\sigma) & = & \mathsf{P}_{\otimes}(\mathsf{reify}_{M,N,\Gamma}(\pi_1 \circ \sigma'), \mathsf{reify}_{N,M,\Gamma}(\pi_2 \circ \sigma')) \\
 & & \text{where } \sigma' = \mathsf{dist}_{-,\Gamma} \circ [\![\Gamma]\!]^{-}(\mathsf{dec}) \circ \sigma \\
\mathsf{reify}_{A \oslash N,\Gamma}(\sigma) & = & \mathsf{P}_{\oslash}(\mathsf{reify}_{A,N,\Gamma}(\sigma)) \\
\mathsf{reify}_{A \lhd P,\Gamma}(\sigma) & = & \mathsf{P}_{\lhd}(\mathsf{reify}_{A,P,\Gamma}(\sigma)) \\
\mathsf{reify}_{\top}(\sigma) & = & \mathsf{P}_{\top} \\
\mathsf{reify}_{\top,P,\Gamma}(\sigma) & = & \mathsf{P}^{-}_{\bot}(\mathsf{reify}_{\bot,\Gamma}(\sigma \circ [\![\Gamma]\!]^{+}(\mathsf{abs}^{-1}))) \\
\mathsf{reify}_{\top,N}(\sigma) & = & \mathsf{P}^{-}_{\top}(\mathsf{reify}_{N}((\_ \multimap \bot)^{-1}(\mathsf{unit}^{-1}_{\multimap} \circ \sigma))) \\
\mathsf{reify}_{\top,N,M,\Gamma}(\sigma) & = & \mathsf{P}^{\otimes}_{\top}(\mathsf{reify}_{\top,N \otimes M,\Gamma}(\sigma \circ [\![\Gamma]\!]^{+}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id})))) \\
\mathsf{reify}_{\top,N,P,\Gamma}(\sigma) & = & \mathsf{P}^{\lhd}_{\top}(\mathsf{reify}_{\top,N \lhd P,\Gamma}(\sigma \circ [\![\Gamma]\!]^{+}(\mathsf{lfe}^{-1}))) \\
\mathsf{reify}_{P \oplus Q,\Gamma}(\sigma) & = & [\mathsf{P}_{\oplus 1} \circ \mathsf{reify}_{P,\Gamma}, \mathsf{P}_{\oplus 2} \circ \mathsf{reify}_{Q,\Gamma}] \circ \mathsf{d}^{-1}(\sigma \circ \mathsf{dist}^{-1}_{+,\Gamma}) \\
\mathsf{reify}_{P \parr Q,\Gamma}(\sigma) & = & [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P,Q,\Gamma}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{Q,P,\Gamma}] \circ \mathsf{d}^{-1}(\sigma \circ [\![\Gamma]\!]^{+}(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{+,\Gamma})
\end{array}
$$

2.4.4 Full Completeness for Core Rules

We prove the following full completeness result.

Theorem 2.4.2 *In any complete WS-category, if* $\sigma : [\![\vdash \Gamma]\!]$ *then* σ *is the denotation of a unique analytic proof* $\mathsf{reify}_{\Gamma}(\sigma) \vdash \Gamma$.

reify_{Γ} is defined inductively in Figure 2-7.

Proposition 2.4.3 reify_{Γ} *is a well-defined, terminating procedure.*

Proof We will define a measure on sequents that strictly decreases in the inductive call. Thus, reify is defined inductively on this measure.

Let $\mathbb{N}$ denote the poset of naturals with the usual ordering, and $\mathbb{N} \cup \{\infty\}$ its extension with ∞ dominating all finite elements in $\mathbb{N}$. The codomain of our measure is $\mathbb{N} \times \mathbb{N} \cup \{\infty\} \times \mathbb{N}$, ordered lexicographically. This is a well-ordered set — there are no infinitely descending chains.

- Let $|\Gamma|$ denote the total size of a sequent. Formally, $|\mathbf{1}| = |\mathbf{0}| = |\bot| = |\top| = 1$, $|A \otimes B| = |A \oslash B| = |A \& B| = |A \lhd B| = |A \parr B| = |A \oplus B| = |A, B| = 1 + |A| + |B|$.
- Let $\mathsf{tl}(A, \Gamma)$ denote the length of Γ as a list if $A \in \{\top, \bot\}$ or ∞ otherwise.

- Let $\mathsf{hd}(A, \Gamma)$ denote $|A|$.

Our procedure is defined lexicographically on $\langle |\Gamma|, \mathsf{tl}(\Gamma), \mathsf{hd}(\Gamma)\rangle$. We can see that each time reify is used recursively, this measure strictly decreases in the lexicographical ordering on $\mathbb{N} \times \mathbb{N} \cup \{\infty\} \times \mathbb{N}$:

- In the case that $\Gamma = \top, N$ or $\bot, P$ the first measure decreases in the call to the inductive hypothesis.
- In other cases where $\Gamma = A, \Gamma'$ with $A \in \{\top, \bot\}$ the first measure does not increase, and the second measure strictly decreases as the tail is shortened in the call to the inductive hypothesis.
- In the case that $\Gamma = A, \Gamma'$ with $A \notin \{\top, \bot\}$ the head formula is decomposed. The first measure does not increase, and either the second or third measure does (the size of the head is decreased, possibly to $\bot$ or $\top$).

For the concrete games model, we could replace the first measure by $\mathsf{depth}(\sigma)$ (i.e. the length of the longest play in σ). Again, this measure strictly decreases if $\Gamma = \bot, P$ or $\top, N$ and does not increase in other cases. ∎

We can complete the proof of Theorem 2.4.2 by showing that reify_Γ gives an inverse to $[\![-]\!]_\Gamma$.

Lemma 2.4.4 *For all* $\sigma : [\![\vdash \Gamma]\!]$ *we have* $[\![\mathsf{reify}_\Gamma(\sigma)]\!] = \sigma$.

Proof We proceed by induction on our reification measure $\langle |\Gamma|, \mathsf{tl}(\Gamma), \mathsf{hd}(\Gamma)\rangle$ using equations that hold in the categorical model. We perform case analysis on Γ.

- If $\Gamma = \mathbf{1}, \Delta$ then both LHS and RHS are mappings from I into the terminal object, hence they must be equal.
- The case $\Gamma = \bot$ is impossible, as by axiom $(1a)$ there are no maps $I \to \bot$.
- If $\Gamma = \bot, N, \Delta$ then $[\![\mathsf{reify}_{\bot,N,\Delta}(\sigma)]\!] = [\![\mathsf{P}^-_\bot(\mathsf{reify}_{\bot,\Delta}([\![\Delta]\!]^-(\mathsf{abs}) \circ \sigma))]\!] = [\![\Delta]\!]^-(\mathsf{abs}^{-1}) \circ [\![\mathsf{reify}_{\bot,\Delta}([\![\Delta]\!]^-(\mathsf{abs}) \circ \sigma)]\!] = [\![\Delta]\!]^-(\mathsf{abs}^{-1}) \circ [\![\Delta]\!]^-(\mathsf{abs}) \circ \sigma = \sigma$ as required.
- If $\Gamma = \bot, P$ then $[\![\mathsf{reify}_{\bot,P}(\sigma)]\!] = [\![\mathsf{P}^+_\bot(\mathsf{reify}_P(\Lambda_I^{-1}(\sigma))]\!] = \Lambda_I([\![\mathsf{reify}_P(\Lambda_I^{-1}(\sigma)]\!])) = \Lambda_I \Lambda_I^{-1}(\sigma) = \sigma$.
- If $\Gamma = \bot, P, N, \Delta$ then $[\![\mathsf{reify}_{\bot,P,N,\Delta}(\sigma)]\!] = [\![\mathsf{P}^{\oslash}_\bot(\mathsf{reify}_{\bot,P\oslash N,\Delta}([\![\Delta]\!]^-(\mathsf{lfe}) \circ \sigma))]\!] = [\![\Delta]\!]^-(\mathsf{lfe}^{-1}) \circ [\![\mathsf{reify}_{\bot,P\oslash N,\Delta}([\![\Delta]\!]^-(\mathsf{lfe}) \circ \sigma)]\!] = [\![\Delta]\!]^-(\mathsf{lfe}^{-1}) \circ [\![\Delta]\!]^-(\mathsf{lfe}) \circ \sigma = \sigma$ as required.

- If $\Gamma = \bot, P, Q, \Delta$ then $[\![\mathsf{reify}_{\bot,P,Q,\Delta}(\sigma)]\!] = [\![\mathsf{P}^{\parr}_{\bot}(\mathsf{reify}_{\bot,P \parr Q,\Delta}([\![\Delta]\!]^{-}((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_{\multimap}) \circ \sigma))]\!] = [\![\Delta]\!]^{-}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id})) \circ [\![\mathsf{reify}_{\bot,P \parr Q,\Delta}([\![\Delta]\!]^{-}((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_{\multimap}) \circ \sigma)]\!] = [\![\Delta]\!]^{-}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id})) \circ [\![\Delta]\!]^{-}((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_{\multimap}) \circ \sigma = \sigma$ as required.

- For $\Gamma = M \& N, \Delta$ we have $[\![\mathsf{reify}_{M\&N,\Delta}(\sigma)]\!] = [\![\mathsf{P}_{\&}(\mathsf{reify}_{M,\Delta}(\pi_1 \circ \mathsf{dist}_{-,\Delta} \circ \sigma), \mathsf{reify}_{N,\Delta}(\pi_2 \circ \mathsf{dist}_{-,\Delta} \circ \sigma))]\!] = \mathsf{dist}^{-1}_{-,\Delta} \circ \langle [\![\mathsf{reify}_{M,\Delta}(\pi_1 \circ \mathsf{dist}_{-,\Delta} \circ \sigma)]\!], [\![\mathsf{reify}_{N,\Delta}(\pi_2 \circ \mathsf{dist}_{-,\Delta} \circ \sigma)]\!] \rangle = \mathsf{dist}^{-1}_{-,\Delta} \circ \langle \pi_1, \pi_2 \rangle \circ \mathsf{dist}_{-,\Delta} \circ \sigma = \mathsf{dist}^{-1}_{-,\Delta} \circ \mathsf{id} \circ \mathsf{dist}_{-,\Delta} \circ \sigma = \sigma$.

- For $\Gamma = M \otimes N, \Delta$ we have $[\![\mathsf{reify}_{M\otimes N,\Delta}(\sigma)]\!] = [\![\mathsf{P}_{\otimes}(\mathsf{reify}_{M,N,\Delta}(\pi_1 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ \sigma), \mathsf{reify}_{N,M,\Delta}(\pi_2 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ \sigma))]\!] = [\![\Delta]\!]^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Delta} \circ \langle [\![\mathsf{reify}_{M,N,\Delta}(\pi_1 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ \sigma)]\!], [\![\mathsf{reify}_{N,M,\Delta}(\pi_2 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ \sigma)]\!] \rangle = [\![\Delta]\!]^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Delta} \circ \langle \pi_1 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ \sigma, \pi_2 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ \sigma \rangle = [\![\Delta]\!]^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Delta} \circ \langle \pi_1, \pi_2 \rangle \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ \sigma = [\![\Delta]\!]^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Delta} \circ \mathsf{id} \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ \sigma = \sigma$ as required.

- If $\Gamma = M \oslash N, \Delta$ then $[\![\mathsf{reify}_{\Gamma}(\sigma)]\!] = [\![\mathsf{P}_{\oslash}(\mathsf{reify}_{M,N,\Delta}(\sigma))]\!] = [\![\mathsf{reify}_{M,N,\Delta}(\sigma)]\!] = \sigma$.

- If $\Gamma = M \lhd P, \Delta$ then $[\![\mathsf{reify}_{\Gamma}(\sigma)]\!] = [\![\mathsf{P}_{\lhd}(\mathsf{reify}_{M,P,\Delta}(\sigma))]\!] = [\![\mathsf{reify}_{M,P,\Delta}(\sigma)]\!] = \sigma$.

- If $\Gamma = \top$ then $\sigma : \bot \to \bot$. But $\mathcal{C}(\bot, \bot) \cong \mathcal{C}(I \multimap \bot, I \multimap \bot) \cong \mathcal{C}(I, I)$ by axiom (2). Hence there is a unique map $\bot \to \bot$ and we must have $\sigma = [\![\mathsf{P}_{\top}]\!]$.

- If $\Gamma = \top, P, \Delta$ then $[\![\mathsf{reify}_{\top,P,\Delta}(\sigma)]\!] = [\![\mathsf{P}^{+}_{\top}(\mathsf{reify}_{\top,\Delta}(\sigma \circ [\![\Delta]\!]^{+}(\mathsf{abs}^{-1})))]\!] = [\![\mathsf{reify}_{\top,\Delta}(\sigma \circ [\![\Delta]\!]^{+}(\mathsf{abs}^{-1}))]\!] \circ [\![\Gamma]\!](\mathsf{abs}) = \sigma \circ [\![\Delta]\!]^{+}(\mathsf{abs}^{-1}) \circ [\![\Delta]\!]^{+}(\mathsf{abs}) = \sigma$ as required.

- If $\Gamma = \top, N$ then $[\![\mathsf{reify}_{\top,N}(\sigma)]\!] = [\![\mathsf{P}^{-}_{\top}(\mathsf{reify}_{N}((\_ \multimap \mathsf{id}_{\bot})^{-1}(\mathsf{unit}^{-1}_{\multimap} \circ \sigma)))]\!] = \mathsf{unit}_{\multimap} \circ ([\![\mathsf{reify}_{N}((\_ \multimap \mathsf{id}_{\bot})^{-1}(\mathsf{unit}^{-1}_{\multimap} \circ \sigma))]\!] \multimap \mathsf{id}) = \mathsf{unit}_{\multimap} \circ ((\_ \multimap \mathsf{id}_{\bot})^{-1}(\mathsf{unit}^{-1}_{\multimap} \circ \sigma) \multimap \mathsf{id}) = \mathsf{unit}_{\multimap} \circ \mathsf{unit}^{-1}_{\multimap} \circ \sigma = \sigma$.

- If $\Gamma = \top, N, P, \Delta$ then $[\![\mathsf{reify}_{\top,N,P,\Delta}(\sigma)]\!] = [\![\mathsf{P}^{\lhd}_{\top}(\mathsf{reify}_{\top,N \lhd P,\Delta}(\sigma \circ [\![\Delta]\!]^{+}(\mathsf{lfe}^{-1})))]\!] = [\![\mathsf{reify}_{\top,N \lhd P,\Delta}(\sigma \circ [\![\Delta]\!]^{+}(\mathsf{lfe}^{-1}))]\!] \circ [\![\Delta]\!]^{+}(\mathsf{lfe}) = \sigma \circ [\![\Delta]\!]^{+}(\mathsf{lfe}^{-1}) \circ [\![\Delta]\!]^{+}(\mathsf{lfe}) = \sigma$ as required.

- If $\Gamma = \top, N, M, \Delta$ then $[\![\mathsf{reify}_{\top,N,M,\Delta}(\sigma)]\!] = [\![\mathsf{P}^{\otimes}_{\top}(\mathsf{reify}_{\top,N\otimes M,\Delta}(\sigma \circ [\![\Delta]\!]^{+}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id}))))]\!] = [\![\mathsf{reify}_{\top,N\otimes M,\Delta}(\sigma \circ [\![\Delta]\!]^{+}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id})))]\!] \circ [\![\Delta]\!]^{+}((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_{\multimap}) = \sigma \circ [\![\Delta]\!]^{+}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id})) \circ [\![\Delta]\!]^{+}((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_{\multimap}) = \sigma$ as required.

- If $\Gamma = P_1 \oplus P_2, \Delta$ then $[\![\mathsf{reify}_{P_1 \oplus P_2,\Delta}(\sigma)]\!] = [\![[\mathsf{P}_{\oplus 1} \circ \mathsf{reify}_{P_1,\Delta}, \mathsf{P}_{\oplus 2} \circ \mathsf{reify}_{P_2,\Delta}] \circ \mathsf{d}^{-1}(\sigma \circ \mathsf{dist}^{-1}_{+,\Delta})]\!]$. Suppose $\mathsf{d}^{-1}(\sigma \circ \mathsf{dist}^{-1}_{+,\Delta}) = \mathsf{in}_i(\tau)$, so $\tau \circ \pi_i = \sigma \circ \mathsf{dist}^{-1}_{+,\Delta}$. Then $[\![[\mathsf{P}_{\oplus 1} \circ \mathsf{reify}_{P_1,\Delta}, \mathsf{P}_{\oplus 2} \circ \mathsf{reify}_{P_2,\Delta}] \circ \mathsf{d}^{-1}(\sigma \circ \mathsf{dist}^{-1}_{+,\Delta})]\!] = [\![\mathsf{P}_{\oplus i}(\mathsf{reify}_{P_i,\Delta}(\tau))]\!] = [\![\mathsf{reify}_{P_i,\Delta}(\tau)]\!] \circ [\![\Delta]\!]^{+}(\pi_i) = \tau \circ \pi_i \circ \mathsf{dist}_{+,\Delta} = \sigma \circ \mathsf{dist}^{-1}_{+,\Delta} \circ \mathsf{dist}_{+,\Delta} = \sigma$.

- If $\Gamma = P_1 \parr P_2, \Delta$ then $[\![\mathsf{reify}_{P_1 \parr P_2,\Delta}(\sigma)]\!] = [\![[\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}(\sigma \circ [\![\Delta]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{+,\Delta})]\!]$. Suppose $\mathsf{d}^{-1}(\sigma \circ [\![\Delta]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{+,\Delta}) = \mathsf{in}_i(\tau)$, so $\tau \circ \pi_i = \sigma \circ [\![\Delta]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{+,\Delta}$.

  If $i = 1$ then $[\![[\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}(\sigma \circ [\![\Delta]\!]^+(\mathsf{dec}) \circ \mathsf{dist}^{-1}_{+,\Delta})]\!] = [\![\mathsf{P}_{\parr 1}(\mathsf{reify}_{P_1,P_2,\Delta}(\tau))]\!] = [\![\mathsf{reify}_{P_1,P_2,\Delta}(\tau)]\!] \circ [\![\Delta]\!]^+(\mathsf{wk}) = [\![\mathsf{reify}_{P_1,P_2,\Delta}(\tau)]\!] \circ [\![\Delta]\!]^+(\pi_1 \circ \mathsf{dec}) = \tau \circ [\![\Delta]\!]^+(\pi_1 \circ \mathsf{dec}) = \tau \circ \pi_1 \circ \mathsf{dist}_{+,\Delta} \circ [\![\Delta]\!]^+(\mathsf{dec}) = \sigma \circ [\![\Delta]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{+,\Delta} \circ \mathsf{dist}_{+,\Delta} \circ [\![\Delta]\!]^+(\mathsf{dec}) = \sigma$.

  If $i = 2$ then $[\![[\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}(\sigma \circ [\![\Delta]\!]^+(\mathsf{dec}) \circ \mathsf{dist}^{-1}_{+,\Delta})]\!] = [\![\mathsf{P}_{\parr 2}(\mathsf{reify}_{P_2,P_1,\Delta}(\tau))]\!] = [\![\mathsf{reify}_{P_2,P_1,\Delta}(\tau)]\!] \circ [\![\Delta]\!]^+(\mathsf{wk} \circ \mathsf{sym}) = [\![\mathsf{reify}_{P_2,P_1,\Delta}(\tau)]\!] \circ [\![\Delta]\!]^+(\pi_2 \circ \mathsf{dec}) = \tau \circ [\![\Delta]\!]^+(\pi_2 \circ \mathsf{dec}) = \tau \circ \pi_2 \circ \mathsf{dist}_{+,\Delta} \circ [\![\Delta]\!]^+(\mathsf{dec}) = \sigma \circ [\![\Delta]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{+,\Delta} \circ \mathsf{dist}_{+,\Delta} \circ [\![\Delta]\!]^+(\mathsf{dec}) = \sigma$.

- If $\Gamma = P \oslash M, \Delta$ then $[\![\mathsf{reify}_{P \oslash M,\Delta}(\sigma)]\!] = [\![\mathsf{P}_{\oslash}(\mathsf{reify}_{P,M,\Delta}(\sigma))]\!] = [\![\mathsf{reify}_{P,M,\Delta}(\sigma)]\!] = \sigma$.

- If $\Gamma = P \lhd Q, \Delta$ then $[\![\mathsf{reify}_{P \lhd Q,\Delta}(\sigma)]\!] = [\![\mathsf{P}_{\lhd}(\mathsf{reify}_{P,Q,\Delta}(\sigma)]\!] = [\![\mathsf{reify}_{P,Q,\Delta}(\sigma)]\!] = \sigma$. ∎

Lemma 2.4.5 *For any analytic proof p of $\vdash \Gamma$ we have* $\mathsf{reify}_\Gamma([\![p]\!]) = p$.

Proof We proceed by induction on p.

- If $p = \mathsf{P}_{\mathbf{1}}$ then our conclusion holds since there is a unique analytic proof of $\vdash \mathbf{1}$. This is also true for $p = \mathsf{P}_\top$.

- If $p = \mathsf{P}^+_\bot(p')$ with $\Gamma = \bot, P$ then $\mathsf{reify}_\Gamma([\![p]\!]) = \mathsf{P}^+_\bot(\mathsf{reify}_P(\Lambda_I^{-1}([\![p]\!])) = \mathsf{P}^+_\bot(\mathsf{reify}_P(\Lambda_I^{-1}\Lambda_I[\![p']\!])) = \mathsf{P}^+_\bot(\mathsf{reify}_P([\![p']\!])) = \mathsf{P}^+_\bot(p') = p$.

- If $p = \mathsf{P}^-_\bot(p')$ with $\Gamma = \bot, M, \Delta$ then $\mathsf{reify}_{\bot,M,\Delta}([\![\mathsf{P}^-_\bot(p')]\!])$
  $= \mathsf{P}^-_\bot(\mathsf{reify}_{\bot,\Delta}([\![\Gamma]\!]^-(\mathsf{abs}) \circ [\![\mathsf{P}^-_\bot(p')]\!]))$
  $= \mathsf{P}^-_\bot(\mathsf{reify}_{\bot,\Delta}([\![\Gamma]\!]^-(\mathsf{abs}) \circ [\![\Gamma]\!]^-(\mathsf{abs}^{-1}) \circ [\![p']\!]))$
  $= \mathsf{P}^-_\bot(\mathsf{reify}_{\bot,\Delta}([\![p']\!]))$
  $= \mathsf{P}^-_\bot(p') = p$ as required.

- If $p = \mathsf{P}^{\parr}_\bot(p')$ with $\Gamma = \bot, P, Q, \Delta$ then $\mathsf{reify}_{\bot,P,Q,\Delta}([\![\mathsf{P}^{\parr}_\bot(p')]\!])$
  $= \mathsf{P}^{\parr}_\bot(\mathsf{reify}_{\bot,P \parr Q,\Delta}([\![\Gamma]\!]^-((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_\multimap) \circ [\![\mathsf{P}^-_\bot(p')]\!]))$
  $= \mathsf{P}^{\parr}_\bot(\mathsf{reify}_{\bot,P \parr Q,\Delta}([\![\Gamma]\!]^-((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_\multimap) \circ [\![\Gamma]\!]^-(\mathsf{pasc}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id})) \circ [\![p']\!]))$
  $= \mathsf{P}^{\parr}_\bot(\mathsf{reify}_{\bot,P \parr Q,\Delta}([\![p']\!])) = \mathsf{P}^{\parr}_\bot(p') = p$ as required.

- If $p = \mathsf{P}^{\oslash}_\bot(p')$ with $\Gamma = \bot, P, N, \Delta$ then $\mathsf{reify}_{\bot,P,N,\Delta}([\![\mathsf{P}^{\oslash}_\bot(p')]\!])$
  $= \mathsf{P}^{\oslash}_\bot(\mathsf{reify}_{\bot,P \oslash N,\Delta}([\![\Gamma]\!]^-(\mathsf{lfe}) \circ [\![\mathsf{P}^-_\bot(p')]\!]))$

$= \mathsf{P}^{\oslash}_{\bot}(\mathsf{reify}_{\bot,P\oslash N,\Delta}([\![\Gamma]\!]^{-}(\mathsf{lfe}) \circ [\![\Gamma]\!]^{-}(\mathsf{lfe}^{-1}) \circ [\![p']\!]))$
$= \mathsf{P}^{\oslash}_{\bot}(\mathsf{reify}_{\bot,P\oslash N,\Delta}([\![p']\!]))$
$= \mathsf{P}^{\oslash}_{\bot}(p') = p$ as required.

- If $p = \mathsf{P}_{\&}(p_1, p_2)$ with $\Gamma = M \& N, \Delta$ then $\mathsf{reify}_{\Gamma}([\![p]\!])$
$= \mathsf{P}_{\&}(\mathsf{reify}_{M,\Delta}(\pi_1 \circ \mathsf{dist}_{-,\Delta} \circ [\![p]\!]), \mathsf{reify}_{N,\Delta}(\pi_2 \circ \mathsf{dist}_{-,\Delta} \circ [\![p]\!]))$
$= \mathsf{P}_{\&}(\mathsf{reify}_{M,\Delta}(\pi_1 \circ \mathsf{dist}_{-,\Delta} \circ \mathsf{dist}^{-1}_{-,\Delta} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle), \mathsf{reify}_{N,\Delta}(\pi_2 \circ \mathsf{dist}_{-,\Delta} \circ \mathsf{dist}^{-1}_{-,\Delta} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle))$
$= \mathsf{P}_{\&}(\mathsf{reify}_{M,\Delta}([\![p_1]\!]), \mathsf{reify}_{N,\Delta}([\![p_2]\!]))$
$= \mathsf{P}_{\&}(p_1, p_2) = p.$

- If $p = \mathsf{P}_{\otimes}(p_1, p_2)$ with $\Gamma = M \otimes N, \Delta$ then $\mathsf{reify}_{\Gamma}([\![p]\!])$
$= \mathsf{P}_{\otimes}(\mathsf{reify}_{M,N,\Delta}(\pi_1 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ [\![p]\!]), \mathsf{reify}_{N,M,\Delta}(\pi_2 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ [\![p]\!]))$
$= \mathsf{P}_{\otimes}(\mathsf{reify}_{M,N,\Delta}(\pi_1 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ [\![\Delta]\!]^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Delta} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle), \mathsf{reify}_{N,M,\Delta}(\pi_2 \circ \mathsf{dist}_{-,\Delta} \circ [\![\Delta]\!]^{-}(\mathsf{dec}) \circ [\![\Delta]\!]^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Delta} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle))$
$= \mathsf{P}_{\otimes}(\mathsf{reify}_{M,\Delta}([\![p_1]\!]), \mathsf{reify}_{N,\Delta}([\![p_2]\!]))$
$= \mathsf{P}_{\otimes}(p_1, p_2) = p.$

- If $p = \mathsf{P}^{-}_{\top}(p')$ and $\Gamma = \top, N$ then $\mathsf{reify}_{\Gamma}([\![p]\!]) = \mathsf{P}^{-}_{\top}(\mathsf{reify}_{N}((\_ \multimap \mathsf{id}_{\bot})^{-1}(\mathsf{unit}^{-1}_{\multimap} \circ [\![p]\!]))) = \mathsf{P}^{-}_{\top}(\mathsf{reify}_{N}((\_ \multimap \mathsf{id}_{\bot})^{-1}(\mathsf{unit}^{-1}_{\multimap} \circ \mathsf{unit}_{\multimap} \circ ([\![p']\!] \multimap \mathsf{id})))) = \mathsf{P}^{-}_{\top}(\mathsf{reify}_{N}((\_ \multimap \mathsf{id}_{\bot})^{-1}([\![p']\!] \multimap \mathsf{id}))) = \mathsf{P}^{-}_{\top}(\mathsf{reify}_{N}[\![p']\!]) = \mathsf{P}^{-}_{\top}(p') = p.$

- If $p = \mathsf{P}^{+}_{\top}(p')$ then $\mathsf{reify}_{\top,P,\Delta}([\![\mathsf{P}^{+}_{\top}(p')]\!]) = \mathsf{P}^{+}_{\top}(\mathsf{reify}_{\top,\Delta}([\![\mathsf{P}^{+}_{\top}(p')]\!] \circ [\![\Delta]\!]^{+}(\mathsf{abs}^{-1}))) = \mathsf{P}^{+}_{\top}(\mathsf{reify}_{\top,\Delta}([\![p']\!] \circ [\![\Delta]\!]^{+}(\mathsf{abs}) \circ [\![\Delta]\!]^{+}(\mathsf{abs}^{-1}))) = \mathsf{P}^{+}_{\top}(\mathsf{reify}_{\top,\Delta}([\![p']\!])) = \mathsf{P}^{+}_{\top}(p') = p$ as required.

- If $p = \mathsf{P}^{\lhd}_{\top}(p')$ then $\mathsf{reify}_{\top,N,P,\Delta}([\![\mathsf{P}^{\lhd}_{\top}(p')]\!]) = \mathsf{P}^{\lhd}_{\top}(\mathsf{reify}_{\top,N\lhd P,\Delta}([\![\mathsf{P}^{\lhd}_{\top}(p')]\!] \circ [\![\Delta]\!]^{+}(\mathsf{lfe}^{-1}))) = \mathsf{P}^{\lhd}_{\top}(\mathsf{reify}_{\top,N\lhd P,\Delta}([\![p']\!] \circ [\![\Delta]\!]^{+}(\mathsf{lfe}) \circ [\![\Delta]\!]^{+}(\mathsf{lfe}^{-1}))) = \mathsf{P}^{\lhd}_{\top}(\mathsf{reify}_{\top,N\lhd P,\Delta}([\![p']\!])) = \mathsf{P}^{\lhd}_{\top}(p') = p$ as required.

- If $p = \mathsf{P}^{\otimes}_{\top}(p')$ then $\mathsf{reify}_{\top,N,M,\Delta}([\![\mathsf{P}^{\otimes}_{\top}(p')]\!]) = \mathsf{P}^{\lhd}_{\top}(\mathsf{reify}_{\top,N\otimes M,\Delta}([\![\mathsf{P}^{\otimes}_{\top}(p')]\!] \circ [\![\Delta]\!]^{+}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id})))) = \mathsf{P}^{\otimes}_{\top}(\mathsf{reify}_{\top,N\otimes M,\Delta}([\![p']\!] \circ [\![\Delta]\!]^{+}(\mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id})) \circ [\![\Delta]\!]^{+}((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}^{-1}_{\multimap}))) = \mathsf{P}^{\otimes}_{\top}(\mathsf{reify}_{\top,N\otimes M,\Delta}([\![p']\!])) = \mathsf{P}^{\otimes}_{\top}(p') = p$ as required.

- If $p = \mathsf{P}_{\oplus i}(p')$ with $\Gamma = P_1 \oplus P_2, \Delta$ then $\mathsf{reify}_{\Gamma}([\![p]\!]) = \mathsf{reify}_{\Gamma}([\![p']\!] \circ [\![\Delta]\!]^{+}(\pi_i)) = [\mathsf{P}_{\oplus 1} \circ \mathsf{reify}_{P_1,\Delta}, \mathsf{P}_{\oplus 2} \circ \mathsf{reify}_{P_2,\Delta}] \circ \mathsf{d}^{-1}([\![p']\!] \circ [\![\Delta]\!]^{+}(\pi_i) \circ \mathsf{dist}^{-1}_{+,\Delta}) = [\mathsf{P}_{\oplus 1} \circ \mathsf{reify}_{P_1,\Delta}, \mathsf{P}_{\oplus 2} \circ \mathsf{reify}_{P_2,\Delta}] \circ \mathsf{d}^{-1}([\![p']\!] \circ \pi_i) = [\mathsf{P}_{\oplus 1} \circ \mathsf{reify}_{P_1,\Delta}, \mathsf{P}_{\oplus 2} \circ \mathsf{reify}_{P_2,\Delta}] \circ \mathsf{d}^{-1}(\mathsf{d}(\mathsf{in}_i([\![p']\!]))) = [\mathsf{P}_{\oplus 1} \circ \mathsf{reify}_{P_1,\Delta}, \mathsf{P}_{\oplus 2} \circ \mathsf{reify}_{P_2,\Delta}] \circ \mathsf{in}_i([\![p']\!]) = \mathsf{P}_{\oplus i} \circ \mathsf{reify}_{P_i,\Delta}([\![p']\!]) = \mathsf{P}_{\oplus i}(p') = p$ as required.

- If $p = \mathsf{P}_{\parr 1}(p')$ with $\Gamma = P_1 \parr P_2, \Delta$ then $\mathsf{reify}_\Gamma([\![p]\!]) = \mathsf{reify}_\Gamma([\![p']\!] \circ [\![\Delta]\!]^+(\mathsf{wk})) = [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}([\![p']\!] \circ [\![\Delta]\!]^+(\mathsf{wk}) \circ [\![\Delta]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{+,\Delta}) = [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}([\![p']\!] \circ [\![\Delta]\!]^+(\pi_1) \circ \mathsf{dist}^{-1}_{+,\Delta}) = [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}([\![p']\!] \circ \pi_1) = [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}(\mathsf{d}(\mathsf{in}_1([\![p']\!]))) = \mathsf{P}_{\parr 1}(\mathsf{reify}_{P_1,P_2,\Delta}([\![p']\!])) = \mathsf{P}_{\parr 1}(p') = p$ as required.

- If $p = \mathsf{P}_{\parr 2}(p')$ with $\Gamma = P_1 \parr P_2, \Delta$ then $\mathsf{reify}_\Gamma([\![p]\!]) = \mathsf{reify}_\Gamma([\![p']\!] \circ [\![\Delta]\!]^+(\mathsf{wk} \circ \mathsf{sym})) = [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}([\![p']\!] \circ [\![\Delta]\!]^+(\mathsf{wk} \circ \mathsf{sym}) \circ [\![\Delta]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{+,\Delta}) = [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}([\![p']\!] \circ [\![\Delta]\!]^+(\pi_2) \circ \mathsf{dist}^{-1}_{+,\Delta}) = [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}([\![p']\!] \circ \pi_2) = [\mathsf{P}_{\parr 1} \circ \mathsf{reify}_{P_1,P_2,\Delta}, \mathsf{P}_{\parr 2} \circ \mathsf{reify}_{P_2,P_1,\Delta}] \circ \mathsf{d}^{-1}(\mathsf{d}(\mathsf{in}_2([\![p']\!]))) = \mathsf{P}_{\parr 2}(\mathsf{reify}_{P_2,P_1,\Delta}([\![p']\!])) = \mathsf{P}_{\parr 2}(p') = p$ as required.

- Finally, if $p = \mathsf{P}_{\oslash}(p')$ then $\mathsf{reify}([\![\mathsf{P}_{\oslash}(p')]\!]) = \mathsf{P}_{\oslash}(\mathsf{reify}([\![p']\!])) = \mathsf{P}_{\oslash}(p') = p$. If $p = \mathsf{P}_{\lhd}(p')$ then $\mathsf{reify}([\![\mathsf{P}_{\lhd}(p')]\!]) = \mathsf{P}_{\lhd}(\mathsf{reify}([\![p']\!])) = \mathsf{P}_{\lhd}(p') = p$ as required. ∎

This completes the proof of Theorem 2.4.2. This theorem yields the following consequences:

Corollary 2.4.6 *In any complete WS-category, morphisms $\mathcal{C}([\![M]\!], [\![N]\!])$ correspond bijectively to analytic proofs of $\vdash N, M^\perp$.*

Proof Such morphisms correspond to $\mathcal{C}(I, [\![M]\!] \multimap [\![N]\!]) = \mathcal{C}(I, [\![N, M^\perp]\!])$. ∎

Corollary 2.4.7 *For each proof $p \vdash \Gamma$, there is a unique analytic proof $p' \vdash \Gamma$ with $[\![p']\!] = [\![p]\!]$. Thus, all non-core proof rules are admissible.*

Proof Let $p \vdash \Gamma$ be a proof. We can construct the proof $p' = \mathsf{reify}([\![p]\!]) \vdash \Gamma$ using only the core rules. By Theorem 2.4.2 p' is the unique analytic proof with $[\![p']\!] = [\![p]\!]$. ∎

Remark The model of WS in $\mathcal{G}_f$ is *equivalence complete* [55] — each arrow on a type object is the denotation of a unique proof, and each object in the model is isomorphic to a type object (each finite tree of plays can be constructed using the lifts and additives).

2.5 Cut Elimination

We have shown that the non-core rules are admissible via a reduction-free evaluation with respect to a particular complete WS-category. However we do not know that such a procedure is sound with respect to any other WS-category. We will address this here in the case of cut elimination, by defining a corresponding syntactic procedure.

2.5.1 Cut Elimination Procedure

We describe a syntactic procedure to transform an analytic proof of $\vdash A, \Gamma, N^\perp$ and an analytic proof of $\vdash N, P$ into an analytic proof of $\vdash A, \Gamma, P$. This is a special case of $\mathsf{P}_{\mathsf{cut}}$, with Δ^+ a single formula P and Γ_1 empty. We proceed by induction. The interesting cases are the lifts: if $A, \Gamma = \perp, Q$ then $\vdash \perp, Q, N^\perp$ must have been concluded from $\vdash Q \parr N^\perp$, i.e. $\vdash Q, N^\perp$ or $\vdash N^\perp, Q$. In the first case we can apply the inductive hypothesis, but in the second case we cannot. We need an auxiliary procedure cut_2 which turns analytic proofs of $\vdash N^\perp, Q$ and $\vdash N, P$ into a analytic proof of $\vdash Q \parr P$ (from which we can deduce $\vdash \perp, Q, P$ as required). If we think of this procedure as a representation of strategy composition, this corresponds to the situation when some player is set to play in N next and so the next observable move could be in Q or P.

In order to do define cut and cut_2, we require further auxiliary procedures:

- A procedure $\mathsf{symP}\parr$ mapping analytic proofs of $\vdash P \parr Q, \Gamma$ to analytic proofs of $\vdash Q \parr P, \Gamma$. All proofs of $\vdash P \parr Q, \Gamma$ must end with $\mathsf{P}_{\parr 1}$ or $\mathsf{P}_{\parr 2}$. Set $\mathsf{symP}\parr(\mathsf{P}_{\parr 1}(p)) = \mathsf{P}_{\parr 2}(p)$ and $\mathsf{symP}\parr(\mathsf{P}_{\parr 2}(p)) = \mathsf{P}_{\parr 1}(p)$.
- A procedure wk_P which takes a proof of $\vdash \Gamma$ and produces a proof of $\vdash \Gamma, P$.
- A procedure rem_0 which takes a proof of $\vdash \Gamma, \mathbf{0}$ and produces a proof of $\vdash \Gamma$. From it we define the procedure $\mathsf{unP}\parr 0$ taking a proof of $\vdash \mathbf{0} \parr P$ and yielding a proof of $\vdash P$ defined by $\mathsf{unP}\parr 0(\mathsf{P}_{\parr 2}(p)) = \mathsf{rem}_0(p)$ (noting that there are no proofs of $\vdash \mathbf{0}, P$ and hence the argument must be of the form $\mathsf{P}_{\parr 1}(p)$).

The procedures can be defined using usual diagrammatic notation. For brevity, we only give this exposition for some representative cases, as given in Figures 2-8 and 2-9. The full procedures are defined using a term notation based on the names of the core proof rules in Figures 2-10, 2-11, 2-12 and 2-13.

Examination of the tensor case shows that the right premise is duplicated at a single step, and so elimination of a cut in WS can take exponential time in the worst case.

A machine-checkable Agda script is available at [18] which formally shows that all cases have been covered.

Termination

We need to justify termination of cut and cut_2. In almost all cases, the inductive call is structurally smaller (i.e. it is called on a subproof of the original proof). The exception is in the case of $\mathsf{cut}(\mathsf{P}^+_\perp(y), g)$, where the inductive call occurs on $\mathsf{wk}_0(y)$. The issue is that $\mathsf{wk}_0(y)$ is not a structurally smaller proof than $\mathsf{P}^+_\perp(y)$. However, we can define a measure which strictly decreases.

Figure 2-8: Diagram notation for cut elimination

$$\mathsf{cut}\ \dfrac{\dfrac{\vdash M, L, \Gamma, N^\perp \qquad \vdash L, M, \Gamma, N^\perp}{\vdash M \otimes L, \Gamma, N^\perp} \qquad \vdash N, P}{\vdash M \otimes L, \Gamma, P}$$

$$\mapsto$$

$$\dfrac{\mathsf{cut}\ \dfrac{\vdash M, L, \Gamma, N^\perp \qquad \vdash N, P}{\vdash M, L, \Gamma, P} \qquad \mathsf{cut}\ \dfrac{\vdash L, M, \Gamma, N^\perp \qquad \vdash N, P}{\vdash L, M, \Gamma, P}}{\vdash M \otimes L, \Gamma, P}$$

$$\mathsf{cut}\ \dfrac{\dfrac{\vdash Q, \Gamma, N^\perp}{\vdash Q \oplus R, \Gamma, N^\perp} \qquad \vdash N, P}{\vdash Q \oplus R, \Gamma, P} \quad \mapsto \quad \mathsf{cut}\ \dfrac{\vdash Q, \Gamma, N^\perp \qquad \vdash N, P}{\dfrac{\vdash Q, \Gamma, P}{\vdash Q \oplus R, \Gamma, P}}$$

$$\mathsf{cut}\ \dfrac{\dfrac{\dfrac{\dfrac{\vdash Q, N^\perp}{\vdash Q \parr N^\perp}}{\vdash \perp, Q \parr N^\perp}}{\vdash \perp, Q, N^\perp} \qquad \vdash N, P}{\vdash \perp, Q, P} \quad \mapsto \quad \mathsf{cut}\ \dfrac{\vdash Q, N^\perp \qquad \vdash N, P}{\dfrac{\dfrac{\dfrac{\vdash Q, P}{\vdash Q \parr P}}{\vdash \perp, Q \parr P}}{\vdash \perp, Q, P}}$$

$$\mathsf{cut}\ \dfrac{\dfrac{\dfrac{\dfrac{\vdash N^\perp, Q}{\vdash Q \parr N^\perp}}{\vdash \perp, Q \parr N^\perp}}{\vdash \perp, Q, N^\perp} \qquad \vdash N, P}{\vdash \perp, Q, P} \quad \mapsto \quad \mathsf{cut}_2\ \dfrac{\vdash N^\perp, Q \qquad \vdash N, P}{\dfrac{\dfrac{\vdash Q \parr P}{\vdash \perp, Q \parr P}}{\vdash \perp, Q, P}}$$

Figure 2-9: Diagram notation for elimination of cut_2

$$\mathsf{cut}_2\ \dfrac{\dfrac{\vdash \top, M \otimes L, \Gamma}{\vdash \top, M, L, \Gamma} \qquad \dfrac{\vdash \bot, M^\bot \parr L^\bot, \Gamma^\bot, P}{\vdash \bot, M^\bot, L^\bot, \Gamma^\bot, P}}{\vdash N^\bot \parr P} \quad \mapsto \quad \mathsf{cut}_2\ \dfrac{\vdash \bot, M^\bot \parr L^\bot, \Gamma^\bot, P \qquad \vdash \top, M \otimes L, \Gamma}{\vdash N^\bot \parr P}$$

$$\mathsf{cut}_2\ \dfrac{\dfrac{\vdash P}{\vdash \bot, P} \qquad \dfrac{\overline{\vdash \top}}{\vdash \top, P}}{\vdash N^\bot \parr P} \quad \mapsto \quad \mathsf{wk}\ \dfrac{\vdash P}{\dfrac{\vdash P, N^\bot}{\vdash N^\bot \parr P}}$$

$$\mathsf{cut}_2\ \dfrac{\dfrac{\dfrac{\vdash M, N^\bot}{\vdash M \lhd N^\bot}}{\vdash \top, M \lhd N^\bot} \qquad \dfrac{\dfrac{\vdash M^\bot, P}{\vdash M^\bot \parr P}}{\vdash \bot, M^\bot \parr P}}{\vdash N^\bot \parr P} \quad \mapsto \quad \mathsf{cut}_2\ \dfrac{\vdash M^\bot, P \qquad \vdash M, N^\bot}{\mathsf{symP}\parr\ \dfrac{\vdash P \parr N^\bot}{\vdash N^\bot \parr P}}$$

$$\mathsf{cut}_2\ \dfrac{\vdash M, L, \Gamma}{\vdash M \oslash L, \Gamma} \qquad \dfrac{\vdash M^\bot, L^\bot, \Gamma^\bot, P}{\vdash M^\bot \lhd L^\bot, \Gamma^\bot, P}}{\vdash N^\bot \parr P} \quad \mapsto \quad \mathsf{cut}_2\ \dfrac{\vdash M, L, \Gamma \qquad \vdash M^\bot, L^\bot, \Gamma^\bot, P}{\vdash N^\bot \parr P}$$

Figure 2-10: Cut Elimination Procedure for Core Rules ($\mathsf{wk}_P :\vdash \Gamma \to \vdash \Gamma, P$)

$\mathsf{wk}_P(\mathsf{P}_{\mathbf{1}})$	$=$	$\mathsf{P}_{\mathbf{1}}$	$\mathsf{wk}_P(\mathsf{P}_{\oslash}(p))$	$=$	$\mathsf{P}_{\oslash}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}_{\lhd}(p)$	$=$	$\mathsf{P}_{\lhd}(\mathsf{wk}_P(p))$	$\mathsf{wk}_P(\mathsf{P}_{\otimes}(p,q))$	$=$	$\mathsf{P}_{\otimes}(\mathsf{wk}_P(p), \mathsf{wk}_P(q))$
$\mathsf{wk}_P(\mathsf{P}_{\parr 1}(p)$	$=$	$\mathsf{P}_{\parr 1}(\mathsf{wk}_P(p))$	$\mathsf{wk}_P(\mathsf{P}_{\parr 2}(p)$	$=$	$\mathsf{P}_{\parr 2}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}_{\&}(p,q))$	$=$	$\mathsf{P}_{\&}(\mathsf{wk}_P(p), \mathsf{wk}_P(q))$	$\mathsf{wk}_P(\mathsf{P}_{\oplus 1}(p)$	$=$	$\mathsf{P}_{\oplus 1}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}_{\oplus 2}(p)$	$=$	$\mathsf{P}_{\oplus 2}(\mathsf{wk}_P(p))$	$\mathsf{wk}_P(\mathsf{P}^-_{\bot}(p)$	$=$	$\mathsf{P}^-_{\bot}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}^+_{\bot}(p))$	$=$	$\mathsf{P}^{\parr}_{\bot}(\mathsf{P}^+_{\bot}(\mathsf{P}_{\parr 1}(\mathsf{wk}_P(p))))$	$\mathsf{wk}_P(\mathsf{P}^{\parr}_{\bot}(p)$	$=$	$\mathsf{P}^{\parr}_{\bot}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}^{\oslash}_{\bot}(p)$	$=$	$\mathsf{P}^{\oslash}_{\bot}(\mathsf{wk}_P(p))$	$\mathsf{wk}_P(\mathsf{P}^-_{\top}(p))$	$=$	$\mathsf{P}^{\lhd}_{\top}(\mathsf{P}^-_{\top}(\mathsf{P}_{\lhd}(\mathsf{wk}_P(p))))$
$\mathsf{wk}_P(\mathsf{P}_{\top})$	$=$	$\mathsf{P}^{\pm}_{\top}(\mathsf{P}_{\top})$	$\mathsf{wk}_P(\mathsf{P}^+_{\top}(p)$	$=$	$\mathsf{P}^{\pm}_{\top}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}^{\otimes}_{\top}(p)$	$=$	$\mathsf{P}^{\otimes}_{\top}(\mathsf{wk}_P(p))$	$\mathsf{wk}_P(\mathsf{P}^{\lhd}_{\top}(p)$	$=$	$\mathsf{P}^{\lhd}_{\top}(\mathsf{wk}_P(p))$

Figure 2-11: Cut Elimination Procedure for Core Rules ($\mathsf{rem0} :\vdash \Gamma, 0 \to \vdash \Gamma$)

$\mathsf{rem}_0(\mathsf{P}_{\mathbf{1}})$	$=$	$\mathsf{P}_{\mathbf{1}}$	$\mathsf{rem}_0(\mathsf{P}_{\oslash}(p))$	$=$	$\mathsf{P}_{\oslash}(\mathsf{rem}_0(p))$
$\mathsf{rem}_0(\mathsf{P}_{\lhd}(p)$	$=$	$\mathsf{P}_{\lhd}(\mathsf{rem}_0(p))$	$\mathsf{rem}_0(\mathsf{P}_{\otimes}(p,q))$	$=$	$\mathsf{P}_{\otimes}(\mathsf{rem}_0(p), \mathsf{rem}_0(q))$
$\mathsf{rem}_0(\mathsf{P}_{\parr 1}(p)$	$=$	$\mathsf{P}_{\parr 1}(\mathsf{rem}_0(p))$	$\mathsf{rem}_0(\mathsf{P}_{\parr 2}(p)$	$=$	$\mathsf{P}_{\parr 2}(\mathsf{rem}_0(p))$
$\mathsf{rem}_0(\mathsf{P}_{\&}(p,q))$	$=$	$\mathsf{P}_{\&}(\mathsf{rem}_0(p), \mathsf{rem}_0(q))$	$\mathsf{rem}_0(\mathsf{P}_{\oplus 1}(p)$	$=$	$\mathsf{P}_{\oplus 1}(\mathsf{rem}_0(p))$
$\mathsf{rem}_0(\mathsf{P}_{\oplus 2}(p)$	$=$	$\mathsf{P}_{\oplus 2}(\mathsf{rem}_0(p))$	$\mathsf{rem}_0(\mathsf{P}^-_{\bot}(p)$	$=$	$\mathsf{P}^-_{\bot}(\mathsf{rem}_0(p))$
$\mathsf{rem}_0(\mathsf{P}^+_{\bot}(p))$		never happens	$\mathsf{rem}_0(\mathsf{P}^-_{\top}(p))$		never happens
$\mathsf{rem}_0(\mathsf{P}^{\oslash}_{\bot}(p)$	$=$	$\mathsf{P}^{\oslash}_{\bot}(\mathsf{rem}_0(p))$	$\mathsf{rem}_0(\mathsf{P}^{\otimes}_{\top}(p)$	$=$	$\mathsf{P}^{\otimes}_{\top}(\mathsf{rem}_0(p))$
$\mathsf{rem}_0(\mathsf{P}^{\parr}_{\bot}(\mathsf{P}^+_{\bot}(\mathsf{P}_{\parr 1}(p))))$	$=$	$\mathsf{P}^{\pm}_{\bot}(\mathsf{rem}_0(p))$	$\mathsf{rem}_0(\mathsf{P}^{\parr}_{\bot}(p))$	$=$	$\mathsf{P}^{\parr}_{\bot}(\mathsf{rem}_0(p))$
$\mathsf{rem}_0(\mathsf{P}^+_{\top}(\mathsf{P}_{\top}))$	$=$	$\mathsf{P}_{\top}$	$\mathsf{rem}_0(\mathsf{P}^+_{\top}(p)$	$=$	$\mathsf{P}^{\pm}_{\top}(\mathsf{rem}_0(p))$
$\mathsf{rem}_0(\mathsf{P}^{\lhd}_{\top}(\mathsf{P}^-_{\top}(\mathsf{P}_{\lhd}(p))))$	$=$	$\mathsf{P}^-_{\top}(\mathsf{rem}_0(p))$	$\mathsf{rem}_0(\mathsf{P}^{\lhd}_{\top}(p))$	$=$	$\mathsf{P}^{\lhd}_{\top}(\mathsf{rem}_0(p))$

Figure 2-12: Cut Elimination Procedure for Core Rules (cut)

A	Γ	$\mathsf{cut} :\vdash A, \Gamma, N^\perp \times \vdash N, P$	$\to$	$\vdash A, \Gamma, P$
$\mathbf{1}$		$\mathsf{cut}(\mathsf{P}_{\mathbf{1}}, g)$	$=$	$\mathsf{P}_{\mathbf{1}}$
$\perp$	ϵ	$\mathsf{cut}(\mathsf{P}^+_\perp(y), g)$	$=$	$\mathsf{P}^+_\perp(\mathsf{unP}\invamp 0(\mathsf{cut}_2(\mathsf{wk}_{\mathbf{0}}(y), g)))$
$\perp$	Q	$\mathsf{cut}(\mathsf{P}^{\invamp}_\perp(\mathsf{P}^+_\perp(\mathsf{P}_{\invamp 1}(y))), g)$	$=$	$\mathsf{P}^{\invamp}_\perp(\mathsf{P}^+_\perp(\mathsf{P}\invamp_1(\mathsf{cut}(y, g))))$
$\perp$	Q	$\mathsf{cut}(\mathsf{P}^{\invamp}_\perp(\mathsf{P}^+_\perp(\mathsf{P}_{\invamp 2}(y))), g)$	$=$	$\mathsf{P}^{\invamp}_\perp(\mathsf{P}^+_\perp(\mathsf{cut}_2(y, g)))$
$\perp$	Q, R, Γ'	$\mathsf{cut}(\mathsf{P}^{\invamp}_\perp(y), g)$	$=$	$\mathsf{P}^{\invamp}_\perp(\mathsf{cut}(y, g))$
$\perp$	Q, M, Γ'	$\mathsf{cut}(\mathsf{P}^{\oslash}_\perp(y), g)$	$=$	$\mathsf{P}^{\oslash}_\perp(\mathsf{cut}(y, g))$
$\perp$	N, Γ'	$\mathsf{cut}(\mathsf{P}^-_\perp(y), g)$	$=$	$\mathsf{P}^-_\perp(\mathsf{cut}(y, g))$
$\top$	ϵ	$\mathsf{cut}(\mathsf{P}^+_\top(\mathsf{P}_\top), g)$	$=$	$\mathsf{P}^+_\top(\mathsf{P}_\top)$
$\top$	M	$\mathsf{cut}(\mathsf{P}^{\triangleleft}_\top(\mathsf{P}^-_\top(\mathsf{P}_\triangleleft(y))), g)$	$=$	$\mathsf{P}^{\triangleleft}_\top(\mathsf{P}^-_\top(\mathsf{P}_\triangleleft(\mathsf{cut}(y, g))))$
$\top$	M, Q, Γ'	$\mathsf{cut}(\mathsf{P}^{\triangleleft}_\top(y), g)$	$=$	$\mathsf{P}^{\triangleleft}_\top(\mathsf{cut}(y, g))$
$\top$	M, L, Γ'	$\mathsf{cut}(\mathsf{P}^{\otimes}_\top(y), g)$	$=$	$\mathsf{P}^{\otimes}_\top(\mathsf{cut}(y, g))$
$\top$	R, Γ'	$\mathsf{cut}(\mathsf{P}^+_\top(y), g)$	$=$	$\mathsf{P}^+_\top(\mathsf{cut}(y, g))$
$\_ \otimes \_$		$\mathsf{cut}(\mathsf{P}_\otimes(y, y'), g)$	$=$	$\mathsf{P}_\otimes(\mathsf{cut}(y, g), \mathsf{cut}(y', g))$
$\_ \invamp \_$		$\mathsf{cut}(\mathsf{P}_{\invamp 1}(y), g)$	$=$	$\mathsf{P}_{\invamp 1}(\mathsf{cut}(y, g))$
$\_ \invamp \_$		$\mathsf{cut}(\mathsf{P}_{\invamp 1}(y), g)$	$=$	$\mathsf{P}_{\invamp 2}(\mathsf{cut}(y, g))$
$\_ \oslash \_$		$\mathsf{cut}(\mathsf{P}_\oslash(y), g)$	$=$	$\mathsf{P}_\oslash(\mathsf{cut}(y, g))$
$\_ \triangleleft \_$		$\mathsf{cut}(\mathsf{P}_\triangleleft(y), g)$	$=$	$\mathsf{P}_\triangleleft(\mathsf{cut}(y, g))$
$\_ \oplus \_$		$\mathsf{cut}(\mathsf{P}_{\oplus 1}(y), g)$	$=$	$\mathsf{P}_{\oplus 1}(\mathsf{cut}(y, g))$
$\_ \oplus \_$		$\mathsf{cut}(\mathsf{P}_{\oplus 2}(y), g)$	$=$	$\mathsf{P}_{\oplus 2}(\mathsf{cut}(y, g))$
$\_ \& \_$		$\mathsf{cut}(\mathsf{P}_\&(y, y'), g)$	$=$	$\mathsf{P}_\&(\mathsf{cut}(y, g), \mathsf{cut}(y', g))$

Figure 2-13: Cut Elimination Procedure for Core Rules (cut_2)

Q	Γ	$\mathsf{cut}_2 :\vdash Q, \Gamma, N^\perp \times \vdash Q^\perp, \Gamma^\perp, P$	$\to$	$\vdash N^\perp \invamp P$
$\top$	ϵ	$\mathsf{cut}_2(\mathsf{P}^+_\top(\mathsf{P}_\top), \mathsf{P}^+_\perp(y'))$	$=$	$\mathsf{P}_{\invamp 2}(\mathsf{wk}_{N^\perp}(y'))$
$\top$	M	$\mathsf{cut}_2(\mathsf{P}^{\triangleleft}_\top(\mathsf{P}^-_\top(\mathsf{P}_\triangleleft y)))(\mathsf{P}^{\invamp}_\perp(\mathsf{P}^+_\perp(\mathsf{P}_{\invamp 1}(y'))))$	$=$	$\mathsf{symP}\invamp(\mathsf{cut}_2(y', y))$
$\top$	M	$\mathsf{cut}_2(\mathsf{P}^{\triangleleft}_\top(\mathsf{P}^-_\top(\mathsf{P}_\triangleleft y)))(\mathsf{P}^{\invamp}_\perp(\mathsf{P}^+_\perp(\mathsf{P}_{\invamp 2}(y'))))$	$=$	$\mathsf{P}_{\invamp 2}(\mathsf{cut}(y', y))$
$\top$	M, L, Γ'	$\mathsf{cut}_2(\mathsf{P}^{\otimes}_\top(y), \mathsf{P}^{\invamp}_\perp(y'))$	$=$	$\mathsf{cut}_2(y, y')$
$\top$	M, R, Γ'	$\mathsf{cut}_2(\mathsf{P}^{\triangleleft}_\top(y), \mathsf{P}^{\oslash}_\perp(y'))$	$=$	$\mathsf{cut}_2(y, y')$
$\top$	R, Γ'	$\mathsf{cut}_2(\mathsf{P}^+_\top(y), \mathsf{P}^-_\perp(y))$	$=$	$\mathsf{cut}_2(y, y')$
$\_ \invamp \_$		$\mathsf{cut}_2(\mathsf{P}_{\invamp 1}(y_1), \mathsf{P}_\otimes(y_2, y_3))$	$=$	$\mathsf{cut}_2(y_1, y_2)$
$\_ \invamp \_$		$\mathsf{cut}_2(\mathsf{P}_{\invamp 2}(y_1), \mathsf{P}_\otimes(y_2, y_3))$	$=$	$\mathsf{cut}_2(y_1, y_3)$
$\_ \triangleleft \_$		$\mathsf{cut}_2(\mathsf{P}_\triangleleft(y_1), \mathsf{P}_\oslash(y_2))$	$=$	$\mathsf{cut}_2(y_1, y_2)$
$\_ \oslash \_$		$\mathsf{cut}_2(\mathsf{P}_\oslash(y_1), \mathsf{P}_\triangleleft(y_2))$	$=$	$\mathsf{cut}_2(y_1, y_2)$
$\_ \oplus \_$		$\mathsf{cut}_2(\mathsf{P}_{\oplus 1}(y_1), \mathsf{P}_\&(y_2, y_3))$	$=$	$\mathsf{cut}_2(y_1, y_2)$
$\_ \oplus \_$		$\mathsf{cut}_2(\mathsf{P}_{\oplus 2}(y_1), \mathsf{P}_\&(y_2, y_3))$	$=$	$\mathsf{cut}_2(y_1, y_3)$

Call a rule a *lift* if it is $\mathsf{P}^+_\bot$ or $\mathsf{P}^-_\top$. By inspection, $\mathsf{wk}_0(y)$ contains precisely the same number of lifts as y. We can see that each of the recursive calls in cut and cut_2 are to a structurally smaller argument, or an argument with a strictly smaller number of lifts. Since if p is structurally smaller than q it contains no more lifts, we can see that $\mathsf{cut}/\mathsf{cut}_2$ terminates by defining it inductively on the lexicographical ordering $\langle l(p), |p| \rangle$ where $l(p)$ is the number of lifts in p and $|p|$ is the size of p.

2.5.2 Soundness

Despite the fact that we are emulating the mechanics of strategy composition in WS, we can show that the procedure is sound with respect to any WS-category. This proof is included here for completeness, but is tedious and routine. The reader may wish to move on to Section 2.5.3.

First, we show that some equations hold in any WS-category, and then go on to prove soundness of wk_P, rem_0, cut and cut_2.

Proposition 2.5.1 *In any WS-category, the following equations hold:*

1. $\mathsf{pasc}_{\multimap} \circ \Lambda_I(f) = \Lambda_I(\Lambda f)$
2. $\mathsf{unit}^{-1}_{\multimap} = \mathsf{pasc}_{\multimap} \circ (\mathsf{lunit}_\otimes \multimap \mathsf{id})$.
3. $\mathsf{unit}_{\multimap} \circ (\Lambda_I f \multimap \mathsf{id}) \circ \mathsf{lfe} = \mathsf{app}_s \circ (\mathsf{id} \oslash f)$
4. $\mathsf{abs} = \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$
5. $\mathsf{abs} \circ \mathsf{wk} \circ (f \otimes g) = f \circ \mathsf{runit}_\otimes \circ (\mathsf{id} \otimes \mathsf{af})$.
6. $(((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap}) \multimap \mathsf{id}) \circ \mathsf{lfe} = \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$
7. $\mathsf{lfe}_{A,I} \circ \mathsf{unit}^{-1}_\oslash = (\mathsf{unit}_{\multimap} \multimap \mathsf{id})$
8. $\mathsf{abs}_I = \mathsf{unit}_\oslash$

Proof 1. $\mathsf{pasc}_{\multimap} \circ \Lambda_I(f)$
$= \Lambda(\Lambda(\mathsf{app} \circ \mathsf{assoc})) \circ \Lambda_I(f)$
$= \Lambda(\Lambda(\mathsf{app} \circ \mathsf{assoc}) \circ (\Lambda(f \circ \mathsf{runit}_\otimes) \otimes \mathsf{id}))$
$= \Lambda(\Lambda(\mathsf{app} \circ \mathsf{assoc} \circ ((\Lambda(f \circ \mathsf{runit}_\otimes) \otimes \mathsf{id}) \otimes \mathsf{id})))$
$= \Lambda(\Lambda(\mathsf{app} \circ (\Lambda(f \circ \mathsf{runit}_\otimes) \otimes \mathsf{id}) \circ \mathsf{assoc}))$
$= \Lambda(\Lambda(f \circ \mathsf{runit}_\otimes \circ \mathsf{assoc}))$
$= \Lambda(\Lambda(f \circ (\mathsf{id} \otimes \mathsf{runit}_\otimes)))$
$= \Lambda(\Lambda(f) \circ \mathsf{runit}_\otimes)$
$= \Lambda_I(\Lambda f)$ as required.

2. $\mathsf{pasc}_{\multimap} \circ (\mathsf{lunit}_{\otimes} \multimap \mathsf{id})$
   $= \Lambda(\Lambda(\mathsf{app} \circ \mathsf{assoc})) \circ (\mathsf{lunit}_{\otimes} \multimap \mathsf{id})$
   $= \Lambda(\Lambda(\mathsf{app} \circ \mathsf{assoc} \circ (((\mathsf{lunit}_{\otimes} \multimap \mathsf{id}) \otimes \mathsf{id}) \otimes \mathsf{id})))$
   $= \Lambda(\Lambda(\mathsf{app} \circ ((\mathsf{lunit}_{\otimes} \multimap \mathsf{id}) \otimes \mathsf{id}) \circ \mathsf{assoc}))$
   $= \Lambda(\Lambda(\mathsf{app} \circ (\Lambda(\mathsf{app} \circ (\mathsf{id} \otimes \mathsf{lunit}_{\otimes})) \otimes \mathsf{id}) \circ \mathsf{assoc}))$
   $= \Lambda(\Lambda(\mathsf{app} \circ (\mathsf{id} \otimes \mathsf{lunit}_{\otimes}) \circ \mathsf{assoc}))$
   $= \Lambda(\Lambda(\mathsf{app} \circ (\mathsf{runit}_{\otimes} \otimes \mathsf{id})))$
   $= \Lambda(\Lambda(\mathsf{app}) \circ \mathsf{runit}_{\otimes})$
   $= \Lambda(\mathsf{runit}_{\otimes}) = \mathsf{unit}_{\multimap}^{-1}$ as required.

3. $\mathsf{unit}_{\multimap} \circ (\Lambda_I(f) \multimap \mathsf{id}) \circ \mathsf{lfe}$
   $= \mathsf{unit}_{\multimap} \circ (\Lambda_I(f) \multimap \mathsf{id}) \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk} \circ (\mathsf{id} \otimes \Lambda_I(f))$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ \mathsf{pasc}^{-1} \circ (\mathsf{id} \oslash \Lambda_I(f)) \circ \mathsf{wk})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash (\mathsf{id} \otimes \Lambda_I(f))) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash (\mathsf{app} \circ (\mathsf{id} \otimes \Lambda_I(f)))) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash (f \circ \mathsf{runit}_{\otimes})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash f) \circ \mathsf{runit}_{\otimes})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ \mathsf{runit}_{\otimes} \circ ((\mathsf{id} \oslash f) \otimes \mathsf{id}))$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ \mathsf{runit}_{\otimes}) \circ (\mathsf{id} \oslash f)$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{app}_s \otimes \mathsf{id})) \circ (\mathsf{id} \oslash f)$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{runit}_{\otimes}) \circ \mathsf{app}_s \circ (\mathsf{id} \oslash f)$
   $= \mathsf{unit}_{\multimap} \circ \mathsf{unit}_{\multimap}^{-1} \circ \mathsf{app}_s \circ (\mathsf{id} \oslash f)$
   $= \mathsf{app}_s \circ (\mathsf{id} \oslash f)$ as required.

4. abs
   $= \mathsf{unit}_{\multimap} \circ (\Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af})) \multimap \mathsf{id}) \circ \mathsf{lfe} \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ (\Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af})) \multimap \mathsf{id}) \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk} \circ (\mathsf{id} \otimes \Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af})))) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app} \circ \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ (\mathsf{id} \oslash \Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af}))) \circ \mathsf{wk}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af})))) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app} \circ (\Lambda(\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af})) \otimes \mathsf{id}) \circ \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af}) \circ \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{runit}_{\otimes} \circ \mathsf{sym} \circ (\mathsf{af} \otimes \mathsf{id})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{lunit}_{\otimes} \circ (\mathsf{af} \otimes \mathsf{id})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
   $= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}_s \circ \mathsf{wk} \circ (\mathsf{unit}_{\oslash} \otimes \mathsf{id}) \circ ((\mathsf{id} \oslash \mathsf{af}) \otimes \mathsf{id})) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$

$= \mathsf{unit}_{\multimap} \circ \Lambda(\mathsf{app}) \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
$= \mathsf{unit}_{\multimap} \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af}) \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id})$
$= \mathsf{unit}_{\multimap} \circ \mathsf{unit}_{\oslash} \circ (\mathsf{unit}_{\multimap}^{-1} \oslash \mathsf{id}) \circ (\mathsf{id} \oslash \mathsf{af})$
$= \mathsf{unit}_{\multimap} \circ \mathsf{unit}_{\multimap}^{-1} \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$
$= \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$ as required.

5. $\mathsf{abs} \circ \mathsf{wk} \circ (f \otimes g) = \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af}) \circ \mathsf{wk} \circ (f \otimes g) = \mathsf{unit}_{\oslash} \circ \mathsf{wk} \circ (f \otimes \mathsf{af} \circ g) = \mathsf{runit}_{\otimes} \circ (f \otimes \mathsf{af}) = f \circ \mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af})$.

6. $(((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_{\multimap}^{-1}) \multimap \mathsf{id}) \circ \mathsf{lfe}$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk} \circ (\mathsf{id} \otimes ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_{\multimap}^{-1})))$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ (\mathsf{id} \oslash ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_{\multimap}^{-1})) \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ (\mathsf{id} \oslash (\mathsf{id} \otimes ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_{\multimap}^{-1}))) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash (((\mathsf{af} \multimap \mathsf{id})) \otimes \mathsf{id})) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ (\mathsf{id} \oslash (\mathsf{id} \otimes \mathsf{unit}_{\multimap}^{-1})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash (\Lambda(\mathsf{app} \circ (\mathsf{id} \otimes \mathsf{af})) \otimes \mathsf{id})) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ (\mathsf{id} \oslash (\mathsf{id} \otimes \mathsf{unit}_{\multimap}^{-1})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash (\mathsf{app} \circ (\mathsf{id} \otimes \mathsf{af}))) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ (\mathsf{id} \oslash (\mathsf{id} \otimes \mathsf{unit}_{\multimap}^{-1})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash (\mathsf{app} \circ (\mathsf{id} \otimes \mathsf{af}) \circ \mathsf{sym} \circ (\mathsf{id} \otimes \mathsf{unit}_{\multimap}^{-1}))) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash (\mathsf{app} \circ (\mathsf{id} \otimes \mathsf{af}) \circ (\mathsf{unit}_{\multimap}^{-1} \otimes \mathsf{id}) \circ \mathsf{sym})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash (\mathsf{app} \circ (\mathsf{unit}_{\multimap}^{-1} \otimes \mathsf{id}) \circ (\mathsf{id} \otimes \mathsf{af}) \circ \mathsf{sym})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash (\mathsf{app} \circ (\Lambda(\mathsf{runit}_{\otimes}) \otimes \mathsf{id}) \circ (\mathsf{id} \otimes \mathsf{af}) \circ \mathsf{sym})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash (\mathsf{runit}_{\otimes} \circ (\mathsf{id} \otimes \mathsf{af}) \circ \mathsf{sym})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ ((\mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})) \oslash \mathsf{id}) \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ \mathsf{wk} \circ ((\mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})) \otimes \mathsf{id}))$
$= \Lambda(\mathsf{app}_s \circ \mathsf{wk}) \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$
$= \Lambda(\mathsf{app}) \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$
$= \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$ as required.

7. $\mathsf{lfe}_{A,I} \circ \mathsf{unit}_{\oslash}^{-1} = (\mathsf{unit}_{\multimap} \multimap \mathsf{id})$. We show that $(\mathsf{unit}_{\multimap}^{-1} \multimap \mathsf{id}) \circ \mathsf{lfe} = \mathsf{unit}_{\oslash}$. Well
$(\mathsf{unit}_{\multimap}^{-1} \multimap \mathsf{id}) \circ \mathsf{lfe}$
$= (\mathsf{unit}_{\multimap}^{-1} \multimap \mathsf{id}) \circ \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk} \circ (\mathsf{id} \otimes \mathsf{unit}_{\multimap}^{-1}))$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ (\mathsf{id} \oslash \mathsf{unit}_{\multimap}^{-1}) \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ (\mathsf{id} \oslash (\mathsf{id} \otimes \mathsf{unit}_{\multimap}^{-1})) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app}) \circ (\mathsf{id} \oslash (\mathsf{unit}_{\multimap}^{-1} \otimes \mathsf{id})) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{app} \circ (\Lambda(\mathsf{runit}_{\otimes}) \otimes \mathsf{id})) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{id} \oslash \mathsf{runit}_{\otimes}) \circ (\mathsf{id} \oslash \mathsf{sym}) \circ \mathsf{pasc}^{-1} \circ \mathsf{wk})$
$= \Lambda(\mathsf{app}_s \circ (\mathsf{unit}_{\oslash} \oslash \mathsf{id}) \circ \mathsf{wk})$

$= \Lambda(\mathsf{app}_s \circ \mathsf{wk} \circ (\mathsf{unit}_{\oslash} \otimes \mathsf{id}))$
$= \Lambda(\mathsf{app}_s \circ \mathsf{wk}) \circ \mathsf{unit}_{\oslash}$
$= \Lambda_s(\mathsf{app}_s) \circ \mathsf{unit}_{\oslash}$
$= \Lambda_s(\Lambda_s^{-1}(\mathsf{id})) \circ \mathsf{unit}_{\oslash}$
$= \mathsf{id} \circ \mathsf{unit}_{\oslash}$
$= \mathsf{unit}_{\oslash}$ as required.

8. $\mathsf{abs}_I = \mathsf{unit}_{\oslash}$. Well $\mathsf{abs}_I = \mathsf{unit}_{\multimap} \circ (\mathsf{dist}^0_{\multimap}{}^{-1} \multimap \mathsf{id}) \circ \mathsf{lfe}_{I,I} \circ (\mathsf{unit}^{-1}_{\multimap} \oslash \mathsf{id}) = \mathsf{unit}_{\multimap} \circ (\mathsf{dist}^0_{\multimap}{}^{-1} \multimap \mathsf{id}) \circ (\mathsf{unit}_{\multimap} \multimap \mathsf{id}) \circ \mathsf{unit}_{\oslash} \circ (\mathsf{unit}^{-1}_{\multimap} \oslash \mathsf{id}) = \mathsf{unit}_{\multimap} \circ (\mathsf{dist}^0_{\multimap}{}^{-1} \multimap \mathsf{id}) \circ (\mathsf{unit}_{\multimap} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ \mathsf{unit}_{\oslash} = \mathsf{unit}_{\multimap} \circ (\mathsf{unit}_{\multimap} \circ \mathsf{dist}^0_{\multimap}{}^{-1} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ \mathsf{unit}_{\oslash} = \mathsf{unit}_{\multimap} \circ \mathsf{id} \circ \mathsf{unit}^{-1}_{\multimap} \circ \mathsf{unit}_{\oslash} = \mathsf{unit}_{\oslash}$ using the fact that $(\mathsf{dist}^0_{\multimap})_I = \mathsf{unit}_{\multimap}$ since both sides are maps into the terminal object. ∎

Proposition 2.5.2 *If p is a proof of $\vdash A, \Gamma$ then in any WS-category $[\![\mathsf{rem}_0(p)]\!] = \mathsf{unit}_{\multimap} \circ [\![p]\!]$ if A is negative and $[\![p]\!] \circ \mathsf{unit}^{-1}_{\oslash}$ if A is positive.*

Proof Induction on p. First, we consider the cases where A is negative:

- If $p = \mathsf{P_1}$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P_1}]\!] = \mathsf{unit}_{\multimap} \circ [\![\mathsf{P_1}]\!]$ as the codomain is the terminal object.

- If $p = \mathsf{P}^-_\perp(p')$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}^-_\perp(\mathsf{rem}_0(p'))]\!] = [\![\Gamma]\!]^-(\mathsf{abs}^{-1}) \circ [\![\mathsf{rem}_0(p')]\!] = [\![\Gamma]\!]^-(\mathsf{abs}^{-1}) \circ \mathsf{unit}_{\multimap} \circ [\![p]\!] = \mathsf{unit}_{\multimap} \circ [\![\Gamma, \mathbf{0}]\!]^-(\mathsf{abs}^{-1}) \circ [\![p']\!] = \mathsf{unit}_{\multimap} \circ [\![p]\!]$. The second case of $p = \mathsf{P}^{\parr}_\perp(p')$ and the case of $\mathsf{P}^{\oslash}_\perp(p')$ are similar, replacing abs^{-1} for the appropriate morphism.

- For the first case of $\mathsf{P}^{\parr}_\perp(p')$, when $\Gamma = \perp, P$ we have $[\![\mathsf{rem}_0(\mathsf{P}^{\parr}_\perp(\mathsf{P}^+_\perp(\mathsf{P}_{\parr 1}(p))))]\!] = [\![\mathsf{P}^+_\perp(\mathsf{rem}_0(p))]\!] = \Lambda_I([\![p]\!] \circ \mathsf{unit}^{-1}_{\oslash})$. We need to show that this is equal to $\mathsf{unit}_{\multimap} \circ [\![\mathsf{P}^{\parr}_\perp(\mathsf{P}^+_\perp(\mathsf{P}_{\parr 1}(p)))]\!] = \mathsf{unit}_{\multimap} \circ \mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I([\![p]\!] \circ \mathsf{wk}) = \mathsf{unit}_{\multimap} \circ \mathsf{pasc}_{\multimap} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ (\mathsf{wk} \multimap \mathsf{id}) \circ \Lambda_I([\![p]\!]) = (\mathsf{unit}^{-1}_{\oslash} \multimap \mathsf{id}) \circ \Lambda_I([\![p]\!]) = \Lambda_I([\![p]\!] \circ \mathsf{unit}^{-1}_{\oslash})$ as required.

- If $p = \mathsf{P}_{\&}(p_1, p_2)$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}_{\&}(\mathsf{rem}_0(p_1), \mathsf{rem}_0(p_2))]\!] = \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle [\![\mathsf{rem}_0(p_1)]\!], [\![\mathsf{rem}_0(p_2)]\!] \rangle = \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle \mathsf{unit}_{\multimap} \circ [\![p_1]\!], \mathsf{unit}_{\multimap} \circ [\![p_2]\!] \rangle = \mathsf{dist}^{-1}_{-,\Gamma} \circ (\mathsf{unit}_{\multimap} \times \mathsf{unit}_{\multimap}) \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = \mathsf{dist}^{-1}_{-,\Gamma} \circ \mathsf{unit}_{\multimap} \circ \mathsf{dist}^{-1}_{\multimap} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = \mathsf{unit}_{\multimap} \circ (\mathsf{id} \multimap \mathsf{dist}^{-1}_{-,\Gamma}) \circ \mathsf{dist}^{-1}_{\multimap} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = \mathsf{unit}_{\multimap} \circ \mathsf{dist}^{-1}_{-,\Gamma,\mathbf{0}} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = \mathsf{unit}_{\multimap} \circ [\![\mathsf{P}_{\&}(p_1, p_2)]\!]$ as required.

- The case of $p = \mathsf{P}_{\otimes}(p_1, p_2)$ is similar. If $p = \mathsf{P}_{\otimes}(p_1, p_2)$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}_{\otimes}(\mathsf{rem}_0(p_1), \mathsf{rem}_0(p_2))]\!] = [\![\Gamma]\!]^-(\mathsf{dec}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle [\![\mathsf{rem}_0(p_1)]\!], [\![\mathsf{rem}_0(p_2)]\!] \rangle = [\![\Gamma]\!]^-(\mathsf{dec}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle \mathsf{unit}_{\multimap} \circ [\![p_1]\!], \mathsf{unit}_{\multimap} \circ [\![p_2]\!] \rangle = [\![\Gamma]\!]^-(\mathsf{dec}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ (\mathsf{unit}_{\multimap} \times \mathsf{unit}_{\multimap}) \circ$

$\langle[\![p_1]\!],[\![p_2]\!]\rangle = [\![\Gamma]\!]^-(\mathsf{dec}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \mathsf{unit}_{\multimap} \circ \mathsf{dist}^{-1}_{\multimap} \circ \langle[\![p_1]\!],[\![p_2]\!]\rangle = [\![\Gamma]\!]^-(\mathsf{dec}) \circ \mathsf{unit}_{\multimap} \circ (\mathsf{id} \multimap \mathsf{dist}^{-1}_{-,\Gamma}) \circ \mathsf{dist}^{-1}_{\multimap} \circ \langle[\![p_1]\!],[\![p_2]\!]\rangle = \mathsf{unit}_{\multimap} \circ (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{dec})) \circ (\mathsf{id} \multimap \mathsf{dist}^{-1}_{-,\Gamma}) \circ \mathsf{dist}^{-1}_{\multimap} \circ \langle[\![p_1]\!],[\![p_2]\!]\rangle = \mathsf{unit}_{\multimap} \circ [\![\Gamma,\mathbf{0}]\!]^-(\mathsf{dec}) \circ \mathsf{dist}^{-1}_{-,\Gamma,\mathbf{0}} \circ \langle[\![p_1]\!],[\![p_2]\!]\rangle = \mathsf{unit}_{\multimap} \circ [\![\mathsf{P}_{\otimes}(p_1,p_2)]\!]$ as required.

- If A is negative and $p = \mathsf{P}_{\oslash}(p')$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}_{\oslash}(\mathsf{rem}_0(p'))]\!] = [\![\mathsf{rem}_0(p')]\!] = \mathsf{unit}_{\multimap} \circ [\![p']\!] = \mathsf{unit}_{\multimap} \circ [\![\mathsf{P}_{\oslash}(p')]\!] = \mathsf{unit}_{\multimap} \circ [\![p]\!]$. The case for $\mathsf{P}_{\triangleleft}(p)$ is similar.

We next consider the cases where A is positive:

- In the case of $p = \mathsf{P}^+_{\top}(\mathsf{P}_{\top})$, then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}_{\top}]\!] = \mathsf{id}_{\bot} = \mathsf{abs} \circ \mathsf{unit}^{-1}_{\oslash} = [\![\mathsf{P}^+_{\top}(\mathsf{P}_{\top})]\!] \circ \mathsf{unit}^{-1}_{\oslash}$ using $\mathsf{abs}_I = \mathsf{unit}_{\oslash}$ from Proposition 2.5.1.

- In the second case of $p = \mathsf{P}^+_{\top}(p')$ then $[\![\mathsf{rem}_0(\mathsf{P}^+_{\top}(p'))]\!] = [\![\mathsf{rem}_0(p')]\!] \circ [\![\Gamma, P]\!]^+(\mathsf{abs}) = [\![p']\!] \circ \mathsf{unit}^{-1}_{\oslash} \circ ([\![\Gamma]\!]^+(\mathsf{abs}) \oslash \mathsf{id}) = [\![p']\!] \circ [\![\Gamma,\mathbf{0}]\!]^+(\mathsf{abs}) \circ \mathsf{unit}^{-1}_{\oslash} = [\![\mathsf{P}^+_{\top}(p')]\!] \circ \mathsf{unit}^{-1}_{\oslash}$ as required. The second case for cases for $\mathsf{P}^{\triangleleft}_{\top}$ and the case of $\mathsf{P}^{\otimes}_{\top}$ are similar, replacing abs by the appropriate morphism.

- In the case of $p = \mathsf{P}^{\triangleleft}_{\top}(\mathsf{P}^-_{\top}(\mathsf{P}_{\triangleleft}(p')))$ we have $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}^-_{\top}(\mathsf{rem}_0(p))]\!] = \mathsf{unit}_{\multimap} \circ ([\![\mathsf{rem}_0(p')]\!] \multimap \mathsf{id}) = \mathsf{unit}_{\multimap} \circ (\mathsf{unit}_{\multimap} \circ [\![p']\!] \multimap \mathsf{id}) = \mathsf{unit}_{\multimap} \circ ([\![p']\!] \multimap \mathsf{id}) \circ (\mathsf{unit}_{\multimap} \multimap \mathsf{id}) = \mathsf{unit}_{\multimap} \circ ([\![p']\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ \mathsf{unit}^{-1}_{\oslash} = [\![p]\!] \circ \mathsf{unit}^{-1}_{\oslash}$ using the fact that $\mathsf{lfe} \circ \mathsf{unit}^{-1}_{\oslash} = (\mathsf{unit}_{\multimap} \multimap \mathsf{id})$ from Proposition 2.5.1.

- If $p = \mathsf{P}_{\oplus i}(p')$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}_{\oplus i}(\mathsf{rem}_0(p'))]\!] = [\![\mathsf{rem}_0(p')]\!] \circ [\![\Gamma]\!]^+(\pi_i) = [\![p']\!] \circ \mathsf{unit}^{-1}_{\oslash} \circ [\![\Gamma]\!]^+(\pi_i) = [\![p']\!] \circ [\![\Gamma,\mathbf{0}]\!]^+(\pi_i) \circ \mathsf{unit}^{-1}_{\oslash} = [\![p]\!] \circ \mathsf{unit}^{-1}_{\oslash}$ as required.

- The case for $p = \mathsf{P}_{\parr i}(p')$ is simpler, replacing π_1 for wk and π_2 for $\mathsf{wk} \circ \mathsf{sym}$.

- If A is positive and $p = \mathsf{P}_{\triangleleft}(p')$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}_{\triangleleft}(\mathsf{rem}_0(p'))]\!] = [\![\mathsf{rem}_0(p')]\!] = [\![p']\!] \circ \mathsf{unit}^{-1}_{\oslash} = [\![p]\!] \circ \mathsf{unit}^{-1}_{\oslash}$ as required. The case for $\mathsf{P}_{\oslash}(p')$ is similar. ∎

Proposition 2.5.3 $[\![\mathsf{unP}\parr 0(p)]\!] = [\![p]\!] \circ \mathsf{lunit}^{-1}_{\otimes}$

Proof We show that $[\![\mathsf{unP}\parr 0]\!] \circ \mathsf{lunit}_{\otimes} = [\![p]\!]$. We must have $p = \mathsf{P}_{\parr 2}(p')$ and the LHS is $[\![\mathsf{unP}\parr 0(\mathsf{P}_{\parr 2}(p'))]\!] \circ \mathsf{lunit}_{\otimes} = [\![\mathsf{rem}_0(p')]\!] \circ \mathsf{lunit}_{\otimes} = [\![p']\!] \circ \mathsf{unit}^{-1}_{\oslash} \circ \mathsf{lunit}_{\otimes} = [\![p']\!] \circ \mathsf{wk} \circ \mathsf{sym} = [\![\mathsf{P}_{\parr 2}(p')]\!]$ as required. ∎

Proposition 2.5.4 *If p is a proof of $\vdash A, \Gamma$ then in any WS-category $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}^+_{\mathsf{wk}}(p)]\!]$. (This is $(\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![p]\!]$ if A is negative and $[\![p]\!] \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$ if A is positive.)*

Proof Induction on p. We first consider the cases where A is negative:

- If $p = \mathsf{P_1}$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P_1}]\!] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![\mathsf{P_1}]\!]$ as the codomain is the terminal object.

- If $p = \mathsf{P}_\perp^-(p')$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}_\perp^-(\mathsf{wk}_P(p'))]\!] = [\![\Gamma, P]\!]^-(\mathsf{abs}^{-1}) \circ [\![\mathsf{wk}_P(p')]\!] = (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{abs}^{-1})) \circ [\![\mathsf{wk}_P(p')]\!] = (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{abs}^{-1})) \circ (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p']\!] = (\mathsf{af} \multimap \mathsf{id}) \circ (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{abs}^{-1})) \circ \mathsf{unit}_\circ^{-1} \circ [\![p']\!] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![\Gamma]\!]^-(\mathsf{abs}^{-1}) \circ [\![p']\!] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![\mathsf{P}_\perp^-(p')]\!]$ as required. The case of $p = \mathsf{P}_\perp^{\otimes}(p')$ and $\mathsf{P}_\perp^{\oslash}(p')$ are similar, replacing abs^{-1} by the appropriate morphism.

- If $p = \mathsf{P}_\perp^+(p')$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}_\perp^{\otimes}(\mathsf{P}_\perp^+(\mathsf{P}_{\otimes 1}(\mathsf{wk}_P(p'))))]\!] = \mathsf{pasc}_\circ \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I([\![p']\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af}) \circ \mathsf{wk}) = \mathsf{pasc}_\circ \circ ((\mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af}) \circ \mathsf{wk} \circ \mathsf{sym}) \multimap \mathsf{id}) \circ \Lambda_I([\![p']\!]) = \mathsf{pasc}_\circ \circ ((\mathsf{unit}_\oslash \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{af} \otimes \mathsf{id}) \multimap \mathsf{id}) \circ \Lambda_I([\![p']\!]) = \mathsf{pasc}_\circ \circ (\mathsf{af} \otimes \mathsf{id} \multimap \mathsf{id}) \circ (\mathsf{lunit}_\otimes \multimap \mathsf{id}) \circ \Lambda_I([\![p']\!]) = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{pasc}_\circ \circ (\mathsf{lunit}_\otimes \multimap \mathsf{id}) \circ \Lambda_I([\![p']\!]) = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ \Lambda_I([\![p']\!]) = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p]\!]$ as required.

- If $p = \mathsf{P}_\&(p_1, p_2)$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}_\&(\mathsf{wk}_P(p_1), \mathsf{wk}_P(p_2))]\!] = \mathsf{dist}_{-,\Gamma,P}^{-1} \circ \langle [\![\mathsf{wk}_P(p_1)]\!], [\![\mathsf{wk}_P(p_2)]\!] \rangle = \mathsf{dist}_{-,\Gamma,P}^{-1} \circ \langle (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p_1]\!], (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p_2]\!] \rangle = \mathsf{dist}_{-,\Gamma,P}^{-1} \circ (((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1}) \times ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1})) \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ \mathsf{dist}_\circ^{-1} \circ (((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1}) \times ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1})) \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1}) \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{af} \multimap \mathsf{id}) \circ (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ \mathsf{unit}_\circ^{-1} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ \mathsf{dist}_{-,\Gamma}^{-1} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p]\!]$ as required.

- The case of $p = \mathsf{P}_\otimes(p_1, p_2)$ is similar. If $p = \mathsf{P}_\otimes(p_1, p_2)$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}_\otimes(\mathsf{wk}_P(p_1), \mathsf{wk}_P(p_2))]\!] = [\![\Gamma, P]\!]^-(\mathsf{dec}) \circ \mathsf{dist}_{-,\Gamma,P}^{-1} \circ \langle [\![\mathsf{wk}_P(p_1)]\!], [\![\mathsf{wk}_P(p_2)]\!] \rangle = [\![\Gamma, P]\!]^-(\mathsf{dec}) \circ \mathsf{dist}_{-,\Gamma,P}^{-1} \circ \langle (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p_1]\!], (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p_2]\!] \rangle = [\![\Gamma, P]\!]^-(\mathsf{dec}) \circ \mathsf{dist}_{-,\Gamma,P}^{-1} \circ (((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1}) \times ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1})) \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = [\![\Gamma, P]\!]^-(\mathsf{dec}) \circ (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ \mathsf{dist}_\circ^{-1} \circ (((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1}) \times ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1})) \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = [\![\Gamma, P]\!]^-(\mathsf{dec}) \circ (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1}) \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = [\![\Gamma, P]\!]^-(\mathsf{dec}) \circ (\mathsf{af} \multimap \mathsf{id}) \circ (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ \mathsf{unit}_\circ^{-1} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = [\![\Gamma, P]\!]^-(\mathsf{dec}) \circ (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ \mathsf{dist}_{-,\Gamma}^{-1} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{dec})) \circ (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ \mathsf{dist}_{-,\Gamma}^{-1} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{af} \multimap \mathsf{id}) \circ (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{dec})) \circ \mathsf{unit}_\circ^{-1} \circ \mathsf{dist}_{-,\Gamma}^{-1} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ \circ [\![\Gamma]\!]^-(\mathsf{dec}) \circ \mathsf{dist}_{-,\Gamma}^{-1} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p]\!]$ as required.

- If A is negative and $p = \mathsf{P}_\oslash(p')$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}_\oslash(\mathsf{wk}_P(p'))]\!] = [\![\mathsf{wk}_P(p')]\!] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p']\!] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![\mathsf{P}_\oslash(p')]\!] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}_\circ^{-1} \circ [\![p]\!]$. The case for $\mathsf{P}_\lhd(p)$ is similar.

We next consider the cases where A is positive:

- If $p = \mathsf{P}_\top$ then $[\![\mathsf{wk}_P(\mathsf{P}_\top)]\!] = [\![\mathsf{P}^+_\top(\mathsf{P}_\top)]\!] = \mathsf{abs} = \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$ by Proposition 2.5.1. This is $[\![\mathsf{P}_\top]\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$ as required.

- If $p = \mathsf{P}^+_\top(p')$ then $[\![\mathsf{wk}_P(\mathsf{P}^+_\top(p'))]\!] = [\![\mathsf{wk}_P(p')]\!] \circ [\![\Gamma, P]\!]^+(\mathsf{abs}) = [\![p']\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af}) \circ ([\![\Gamma]\!]^+(\mathsf{abs}) \oslash \mathsf{id}) = [\![p']\!] \circ [\![\Gamma]\!]^+(\mathsf{abs}) \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af}) = [\![\mathsf{P}^+_\top(p')]\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$ as required. The cases for $\mathsf{P}^\otimes_\top$ and $\mathsf{P}^\lhd_\top$ are similar, replacing abs by the appropriate morphism.

- If $p = \mathsf{P}^-_\top(p')$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}^\lhd_\top(\mathsf{P}^-_\top(\mathsf{P}_\lhd(\mathsf{wk}_P(p'))))]\!] = \mathsf{unit}_{\multimap} \circ ((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![p']\!]) \multimap \mathsf{id}) \circ \mathsf{lfe} = \mathsf{unit}_{\multimap} \circ ([\![p']\!] \multimap \mathsf{id}) \circ (((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap}) \multimap \mathsf{id}) \circ \mathsf{lfe} = \mathsf{unit}_{\multimap} \circ ([\![p']\!] \multimap \mathsf{id}) \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$ by Proposition 2.5.1. This is $[\![p]\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$ as required.

- If $p = \mathsf{P}_{\oplus i}(p')$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}_{\oplus i}(\mathsf{wk}_P(p'))]\!] = [\![\mathsf{wk}_P(p')]\!] \circ [\![\Gamma, P]\!]^+(\pi_i) = [\![p']\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af}) \circ ([\![\Gamma]\!]^+(\pi_i) \oslash \mathsf{id}) = [\![p']\!] \circ [\![\Gamma]\!]^+(\pi_i) \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af}) = [\![p]\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$ as required.

- The case for $p = \mathsf{P}_{\parr i}(p')$ is simpler, replacing π_1 for wk and π_2 for $\mathsf{wk} \circ \mathsf{sym}$.

- If A is positive and $p = \mathsf{P}_\lhd(p')$ then $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}_\lhd(\mathsf{wk}_P(p'))]\!] = [\![\mathsf{wk}_P(p')]\!] = [\![p']\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af}) = [\![p]\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af})$ as required. The case for $\mathsf{P}_\oslash(p')$ is similar. ∎

We can now show the main result of this section, i.e. that $[\![\mathsf{cut}(p_1, p_2)]\!] = [\![\mathsf{P}_{\mathsf{cut}}(p_1, p_2)]\!]$.

Proposition 2.5.5 *In any WS-category, if p_1 is a proof of $\vdash A, \Gamma, N^\perp$ and p_2 is a proof of $\vdash N, R$ then $[\![\mathsf{cut}(p_1, p_2)]\!] = [\![\mathsf{P}_{\mathsf{cut}}(p_1, p_2)]\!]$. That is,*

- $[\![\mathsf{cut}(p_1, p_2)]\!] = \Lambda_I(\Lambda_I^{-1}([\![p_1]\!]) \circ \Lambda_I^{-1}([\![p_2]\!]))$ *if A is negative.*

- $[\![\mathsf{cut}(p_1, p_2)]\!] = [\![p_1]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![p_2]\!])$ *if A is positive.*

- $[\![\mathsf{cut}_2(p_1, p_2)]\!] = [\![p_1]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![p_2]\!])$.

Proof We show these three facts by simultaneous induction, following the cases defined in the definition of cut and cut_2.

We first show that $[\![\mathsf{cut}(p_1, p_2)]\!] = \Lambda_I(\Lambda_I^{-1}([\![p_1]\!]) \circ \Lambda_I^{-1}([\![p_2]\!]))$ if A is negative.

- If $A = \mathbf{1}$ then the hom-set in question is singleton, since the codomain is isomorphic to the terminal object, hence any two elements of this hom set are equal, in particular $[\![\mathsf{cut}(\mathsf{P}_{\mathbf{1}}, p_2)]\!] = \Lambda_I(\Lambda_I^{-1}([\![\mathsf{P}_{\mathbf{1}}]\!]) \circ \Lambda_I^{-1}([\![p_2]\!]))$.

- If $A = \bot$ and $\Gamma = \epsilon$ then $[\![\mathsf{cut}(\mathsf{P}^+_\bot(p), h)]\!]$
  $= [\![\mathsf{P}^+_\bot(\mathsf{unP}\invamp 0(\mathsf{cut}_2(\mathsf{wk}_\mathbf{0}(p), h)))]\!]$
  $= \Lambda_I([\![\mathsf{cut}_2(\mathsf{wk}_\mathbf{0}(p), h)]\!] \circ \mathsf{lunit}^{-1}_\otimes)$ by Proposition 2.5.3
  $= \Lambda_I([\![\mathsf{wk}_\mathbf{0}(p)]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![h]\!]) \circ \mathsf{lunit}^{-1}_\otimes)$ by Proposition 2.5.4
  $= \Lambda_I([\![p]\!] \circ \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{af}_\mathbf{0}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![h]\!]) \circ \mathsf{lunit}^{-1}_\otimes)$
  $= \Lambda_I([\![p]\!] \circ \mathsf{unit}_\oslash \circ \mathsf{id} \circ \mathsf{wk} \circ \mathsf{sym} \circ \mathsf{lunit}^{-1}_\otimes \circ \Lambda_I^{-1}([\![h]\!]))$
  $= \Lambda_I([\![p]\!] \circ \Lambda_I^{-1}([\![h]\!]))$
  $= \Lambda_I(\Lambda_I^{-1}(\Lambda_I([\![p]\!])) \circ \Lambda_I^{-1}([\![h]\!]))$
  $= \Lambda_I(\Lambda_I^{-1}([\![\mathsf{P}^+_\bot(p)]\!]) \circ \Lambda_I^{-1}([\![h]\!]))$ as required.

- If $A = \bot$ and $\Gamma = P, Q, \Gamma'$ then $[\![\mathsf{cut}(\mathsf{P}^\invamp_\bot(p), h)]\!]$
  $= [\![\mathsf{P}^\invamp_\bot(\mathsf{cut}(p, h))]\!]$
  $= [\![\Gamma, R]\!]^-(\mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id})) \circ [\![\mathsf{cut}(p, h)]\!]$
  $= [\![\Gamma, R]\!]^-(\mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id})) \circ \Lambda_I(\Lambda_I^{-1}[\![p]\!] \circ \Lambda_I^{-1}[\![h]\!])$
  $= (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id}))) \circ \Lambda_I(\Lambda_I^{-1}[\![p]\!] \circ \Lambda_I^{-1}[\![h]\!])$
  $= \Lambda_I([\![\Gamma]\!]^-(\mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id})) \circ \Lambda_I^{-1}[\![p]\!] \circ \Lambda_I^{-1}[\![h]\!])$
  $= \Lambda_I(\Lambda_I^{-1}((\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id}))) \circ [\![p]\!]) \circ \Lambda_I^{-1}[\![h]\!])$
  $= \Lambda_I(\Lambda_I^{-1}([\![\Gamma, N^\bot]\!]^-(\mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id})) \circ [\![p]\!]) \circ \Lambda_I^{-1}[\![h]\!])$
  $= \Lambda_I(\Lambda_I^{-1}[\![\mathsf{P}^\invamp_\bot(p)]\!] \circ \Lambda_I^{-1}[\![h]\!])$ as required.

- If $A = \bot$ and $\Gamma = P, M, \Gamma'$ then the proof is entirely similar, replacing lfe^{-1} for $\mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id})$.

- If $A = \bot$ and $\Gamma = M, \Gamma$ then the proof is entirely similar, replacing lfe^{-1} for the isomorphism abs.

- If $A = \bot$ and $\Gamma = P$ then $[\![\mathsf{cut}(\mathsf{P}^\invamp_\bot(\mathsf{P}^+_\bot(\mathsf{P}_{\invamp 1}(y))), g)]\!]$
  $= [\![\mathsf{P}^\invamp_\bot(\mathsf{P}^+_\bot(\mathsf{P}_{\invamp 1}(\mathsf{cut}(y, g))))]\!]$
  $= \mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I([\![y]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) \circ \mathsf{wk})$
  $= \mathsf{pasc}^{-1}_\multimap \circ \Lambda_I([\![y]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) \circ \mathsf{wk} \circ \mathsf{sym})$
  $= \Lambda_I(\Lambda([\![y]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) \circ \mathsf{wk} \circ \mathsf{sym}))$ using Proposition 2.5.1
  $= \Lambda_I\Lambda([\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\Lambda_I^{-1}[\![g]\!] \otimes \mathsf{id}))$
  $= \Lambda_I(\Lambda([\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym}) \circ \Lambda_I^{-1}[\![g]\!])$
  $= \Lambda_I(\Lambda_I^{-1}\Lambda_I\Lambda([\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym}) \circ \Lambda_I^{-1}[\![g]\!])$
  $= \Lambda_I(\Lambda_I^{-1}(\mathsf{pasc}^{-1}_\multimap \circ \Lambda_I([\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym})) \circ \Lambda_I^{-1}[\![g]\!])$ using Proposition 2.5.1
  $= \Lambda_I(\Lambda_I^{-1}(\mathsf{pasc}^{-1}_\multimap \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I([\![y]\!] \circ \mathsf{wk})) \circ \Lambda_I^{-1}[\![g]\!])$
  $= \Lambda_I(\Lambda_I^{-1}[\![\mathsf{P}^\invamp_\bot(\mathsf{P}^+_\bot(\mathsf{P}_{\invamp 1}(y)))]\!] \circ \Lambda_I^{-1}[\![g]\!])$ as required.

  Similarly, $[\![\mathsf{cut}(\mathsf{P}^\invamp_\bot(\mathsf{P}^+_\bot(\mathsf{P}_{\invamp 2}(y))), g)]\!]$
  $= [\![\mathsf{P}^\invamp_\bot(\mathsf{P}^+_\bot(\mathsf{cut}_2(y, g)))]\!]$

$= \mathsf{pasc}_{\multimap}^{-1} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I(\llbracket y \rrbracket \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}\llbracket g \rrbracket))$
$= \mathsf{pasc}_{\multimap}^{-1} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I(\llbracket y \rrbracket \circ \mathsf{wk} \circ (\Lambda_I^{-1}\llbracket g \rrbracket \otimes \mathsf{id}) \circ \mathsf{sym})$
$= \mathsf{pasc}_{\multimap}^{-1} \circ \Lambda_I(\llbracket y \rrbracket \circ \mathsf{wk} \circ (\Lambda_I^{-1}\llbracket g \rrbracket \otimes \mathsf{id}) \circ \mathsf{sym} \circ \mathsf{sym})$
$= \mathsf{pasc}_{\multimap}^{-1} \circ \Lambda_I(\llbracket y \rrbracket \circ \mathsf{wk} \circ (\Lambda_I^{-1}\llbracket g \rrbracket \otimes \mathsf{id}))$
$= \Lambda_I\Lambda(\llbracket y \rrbracket \circ \mathsf{wk} \circ (\Lambda_I^{-1}\llbracket g \rrbracket \otimes \mathsf{id}))$ using Proposition 2.5.1
$= \Lambda_I(\Lambda(\llbracket y \rrbracket \circ \mathsf{wk}) \circ \Lambda_I^{-1}(\llbracket g \rrbracket))$
$= \Lambda_I(\Lambda_I^{-1}\Lambda_I\Lambda(\llbracket y \rrbracket \circ \mathsf{wk}) \circ \Lambda_I^{-1}(\llbracket g \rrbracket))$
$= \Lambda_I(\Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ \Lambda_I(\llbracket y \rrbracket \circ \mathsf{wk}) \circ \Lambda_I^{-1}(\llbracket g \rrbracket))$ using Proposition 2.5.1
$= \Lambda_I(\Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ \Lambda_I(\llbracket y \rrbracket \circ \mathsf{wk} \circ \mathsf{sym} \circ \mathsf{sym}) \circ \Lambda_I^{-1}(\llbracket g \rrbracket))$
$= \Lambda_I(\Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I(\llbracket y \rrbracket \circ \mathsf{wk} \circ \mathsf{sym}) \circ \Lambda_I^{-1}(\llbracket g \rrbracket))$
$= \Lambda_I(\Lambda_I^{-1}(\llbracket \mathsf{P}_{\perp}^{\parr}(\mathsf{P}_{\perp}^{+}(\mathsf{P}_{\parr 2}(y))) \rrbracket) \circ \Lambda_I^{-1}(\llbracket g \rrbracket))$ as required.

- If $A = M \& N$ then $\llbracket \mathsf{cut}(\mathsf{P}_{\&}(p_a, p_b), g) \rrbracket = \llbracket \mathsf{P}_{\&}(\mathsf{cut}(p_a, g), \mathsf{cut}(p_b, g)) \rrbracket$
$= \mathsf{dist}_{-,\Gamma,R}^{-1} \circ \langle \llbracket \mathsf{cut}(p_a, g) \rrbracket, \llbracket \mathsf{cut}(p_b, g) \rrbracket \rangle$
$= \mathsf{dist}_{-,\Gamma,R}^{-1} \circ \langle \Lambda_I(\Lambda_I^{-1}\llbracket p_a \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket), \Lambda_I(\Lambda_I^{-1}\llbracket p_b \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket) \rangle$
$= \mathsf{dist}_{-,\Gamma,R}^{-1} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ \Lambda_I\langle \Lambda_I^{-1}\llbracket p_a \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket \rangle$
$= \mathsf{dist}_{-,\Gamma,R}^{-1} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ \Lambda_I\langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle \circ \Lambda_I^{-1}\llbracket g \rrbracket$
$= \mathsf{dist}_{-,\Gamma,R}^{-1} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ (\mathsf{id} \multimap \langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \Lambda_I\Lambda_I^{-1}\llbracket g \rrbracket$
$= \mathsf{dist}_{-,\Gamma,R}^{-1} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ (\mathsf{id} \multimap \langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \llbracket g \rrbracket$
$= (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ (\mathsf{id} \multimap \langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \llbracket g \rrbracket$
$= (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ \langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \llbracket g \rrbracket$
$= (\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ \langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \Lambda_I\Lambda_I^{-1}\llbracket g \rrbracket$
$= \Lambda_I(\mathsf{dist}_{-,\Gamma}^{-1} \circ \langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \Lambda_I^{-1}\llbracket g \rrbracket)$
$= \Lambda_I(\mathsf{dist}_{-,\Gamma}^{-1} \circ \Lambda_I^{-1}\Lambda_I\langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \Lambda_I^{-1}\llbracket g \rrbracket)$
$= \Lambda_I(\Lambda_I^{-1}((\mathsf{id} \multimap \mathsf{dist}_{-,\Gamma}^{-1}) \circ \Lambda_I\langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \Lambda_I^{-1}\llbracket g \rrbracket)$
$= \Lambda_I(\Lambda_I^{-1}(\mathsf{dist}_{-,\Gamma,N}^{-1} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ \Lambda_I\langle \Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \Lambda_I^{-1}\llbracket g \rrbracket)$
$= \Lambda_I(\Lambda_I^{-1}(\mathsf{dist}_{-,\Gamma,N}^{-1} \circ \langle \Lambda_I\Lambda_I^{-1}\llbracket p_a \rrbracket, \Lambda_I\Lambda_I^{-1}\llbracket p_b \rrbracket \rangle) \circ \Lambda_I^{-1}\llbracket g \rrbracket)$
$= \Lambda_I(\Lambda_I^{-1}(\mathsf{dist}_{-,\Gamma,N}^{-1} \circ \langle \llbracket p_a \rrbracket, \llbracket p_b \rrbracket \rangle) \circ \Lambda_I^{-1}\llbracket g \rrbracket)$
$= \Lambda_I(\Lambda_I^{-1}\llbracket \mathsf{P}_{\&}(p_a, p_b) \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket)$ as required.

- If $A = M \oslash N$ then $\llbracket \mathsf{cut}(\mathsf{P}_{\oslash}(p), g) \rrbracket = \llbracket \mathsf{P}_{\oslash}(\mathsf{cut}(p, g)) \rrbracket = \llbracket \mathsf{cut}(p, g) \rrbracket = \Lambda_I(\Lambda_I^{-1}\llbracket p \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket) = \Lambda_I(\Lambda_I^{-1}\llbracket \mathsf{P}_{\oslash}(p) \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket)$ as required.

- If $A = M \lhd P$ then $\llbracket \mathsf{cut}(\mathsf{P}_{\lhd}(p), g) \rrbracket = \llbracket \mathsf{P}_{\lhd}(\mathsf{cut}(p, g)) \rrbracket = \llbracket \mathsf{cut}(p, g) \rrbracket = \Lambda_I(\Lambda_I^{-1}\llbracket p \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket) = \Lambda_I(\Lambda_I^{-1}\llbracket \mathsf{P}_{\lhd}(p) \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket)$ as required.

- If $A = M \otimes N$ then $\llbracket \mathsf{cut}(\mathsf{P}_{\otimes}(p_a, p_b), g) \rrbracket = \llbracket \mathsf{P}_{\otimes}(\mathsf{cut}(p_a, g), \mathsf{cut}(p_b, g)) \rrbracket$
$= \llbracket \Gamma, R \rrbracket^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{-,\Gamma,R}^{-1} \circ \langle \llbracket \mathsf{cut}(p_a, g), \mathsf{cut}(p_b, g) \rangle$
$= \llbracket \Gamma, R \rrbracket^{-}(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{-,\Gamma,R}^{-1} \circ \langle \Lambda_I(\Lambda_I^{-1}\llbracket p_a \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket), \Lambda_I(\Lambda_I^{-1}\llbracket p_b \rrbracket \circ \Lambda_I^{-1}\llbracket g \rrbracket) \rangle$

$= [\![\Gamma, R]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma,R} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ \Lambda_I \langle \Lambda_I^{-1}[\![p_a]\!] \circ \Lambda_I^{-1}[\![g]\!], \Lambda_I^{-1}[\![p_b]\!] \circ \Lambda_I^{-1}[\![g]\!] \rangle$
$= [\![\Gamma, R]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma,R} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ \Lambda_I \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle \circ \Lambda_I^{-1}[\![g]\!]$
$= [\![\Gamma, R]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma,R} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ (\mathsf{id} \multimap \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ \Lambda_I \Lambda_I^{-1}[\![g]\!]$
$= [\![\Gamma, R]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma,R} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ (\mathsf{id} \multimap \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ [\![g]\!]$
$= [\![\Gamma, R]\!]^-(\mathsf{dec}^{-1}) \circ (\mathsf{id} \multimap \mathsf{dist}^{-1}_{-,\Gamma}) \circ (\mathsf{id} \multimap \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ [\![g]\!]$
$= (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{dec}^{-1})) \circ (\mathsf{id} \multimap \mathsf{dist}^{-1}_{-,\Gamma}) \circ (\mathsf{id} \multimap \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ [\![g]\!]$
$= (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma}) \circ (\mathsf{id} \multimap \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ [\![g]\!]$
$= (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ [\![g]\!]$
$= (\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ \Lambda_I \Lambda_I^{-1}[\![g]\!]$
$= \Lambda_I([\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ \Lambda_I^{-1}[\![g]\!])$
$= \Lambda_I([\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \Lambda_I^{-1} \Lambda_I \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ \Lambda_I^{-1}[\![g]\!])$
$= \Lambda_I(\Lambda_I^{-1}((\mathsf{id} \multimap [\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma}) \circ \Lambda_I \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ \Lambda_I^{-1}[\![g]\!])$
$= \Lambda(\Lambda_I^{-1}([\![\Gamma, N]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma,N} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ \Lambda_I \langle \Lambda_I^{-1}[\![p_a]\!], \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ \Lambda_I^{-1}[\![g]\!])$
$= \Lambda_I(\Lambda_I^{-1}([\![\Gamma, N]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma,N} \circ \langle \Lambda_I \Lambda_I^{-1}[\![p_a]\!], \Lambda_I \Lambda_I^{-1}[\![p_b]\!] \rangle) \circ \Lambda_I^{-1}[\![g]\!])$
$= \Lambda_I(\Lambda_I^{-1}([\![\Gamma, N]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma,N} \circ \langle [\![p_a]\!], [\![p_b]\!] \rangle) \circ \Lambda_I^{-1}[\![g]\!])$
$= \Lambda_I(\Lambda_I^{-1}[\![\mathsf{P}_\otimes(p_a, p_b)]\!] \circ \Lambda_I^{-1}[\![g]\!])$ as required.

We next show that $[\![\mathsf{cut}(p_1, p_2)]\!] = [\![p_1]\!] \circ (\mathsf{id}_\Gamma \oslash \Lambda_I^{-1}[\![p_2]\!])$ if A is positive.

- If $A = P \oplus Q$ then $[\![\mathsf{cut}(\mathsf{P}_{\oplus 1}(y), g)]\!] = [\![\mathsf{P}_{\oplus 1}(\mathsf{cut}(y, g))]\!] = [\![\mathsf{cut}(y, g)]\!] \circ [\![\Gamma, R]\!]^+(\pi_1) = [\![y]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) \circ ([\![\Gamma]\!]^+(\pi_1) \oslash \mathsf{id}) = [\![y]\!] \circ ([\![\Gamma]\!]^+(\pi_1) \oslash \Lambda_I^{-1}[\![g]\!]) = [\![y]\!] \circ ([\![\Gamma]\!]^+(\pi_1) \oslash \mathsf{id}) \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) = [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\pi_1) \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) = [\![\mathsf{P}_{\oslash 1}(y)]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!])$ as required. Case for $[\![\mathsf{cut}(\mathsf{P}_{\oplus 2}(y), g)]\!]$ is similar.

- If $A = P \oslash M$ then $[\![\mathsf{cut}(\mathsf{P}_\oslash(y), g)]\!] = [\![\mathsf{P} \oslash (\mathsf{cut}(y, g))]\!] = [\![\mathsf{cut}(y, g)]\!] = [\![y]\!] \circ (\mathsf{id} \oslash [\![g]\!]) = [\![\mathsf{P}_\oslash(y)]\!] \circ (\mathsf{id} \oslash [\![g]\!])$. If $A = P \lhd Q$ similar reasoning applies.

- If $A = P \parr Q$ then $[\![\mathsf{cut}(\mathsf{P}_{\parr 1}(y), g)]\!] = [\![\mathsf{P}_{\parr 1}(\mathsf{cut}(y, g))]\!] = [\![\mathsf{cut}(y, g)]\!] \circ [\![\Gamma, \Delta]\!]^+(\mathsf{wk}) = [\![y]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) \circ ([\![\Gamma]\!]^+(\mathsf{wk}) \oslash \mathsf{id}) = [\![y]\!] \circ ([\![\Gamma]\!]^+(\mathsf{wk}) \oslash \Lambda_I^{-1}[\![g]\!]) = [\![y]\!] \circ ([\![\Gamma]\!]^+(\mathsf{wk}) \oslash \mathsf{id}) \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) = [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\mathsf{wk}) \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) = [\![\mathsf{P}_{\parr 1}(y)]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!])$ as required. Case for $[\![\mathsf{cut}(\mathsf{P}_{\parr 2}(y), g)]\!]$ is similar.

- If $A = \top$ and $\Gamma = N, L, \Gamma'$ then $[\![\mathsf{cut}(\mathsf{P}^\otimes_\top(y), g)]\!] = [\![\mathsf{P}^\otimes_\perp(\mathsf{cut}(p_a, p_2))]\!] = [\![\mathsf{cut}(y, g)]\!] \circ [\![\Gamma, \Delta]\!]^+((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}_\multimap) = [\![y]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) \circ ([\![\Gamma]\!]^+((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}_\multimap) \oslash \mathsf{id}) = [\![y]\!] \circ ([\![\Gamma]\!]^+((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}_\multimap) \oslash \Lambda_I^{-1}[\![g]\!]) = [\![y]\!] \circ ([\![\Gamma]\!]^+((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}_\multimap) \oslash \mathsf{id}) \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) = [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+((\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}_\multimap) \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!]) =$

$[\![\mathsf{P}^{\otimes}_{\bot}(y)]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![g]\!])$ as required. Cases for $\Gamma = N, P, \Gamma'$ and $\Gamma = P, \Gamma'$ are similar, replacing $(\mathsf{sym} \multimap \mathsf{id}) \circ \mathsf{pasc}_{\multimap}$ with the appropriate morphism.

- If $A = \top$ and $\Gamma = N$ then $[\![\mathsf{cut}(\mathsf{P}^{\lhd}_{\top}(\mathsf{P}^{-}_{\top}(\mathsf{P}_{\lhd}(y))), g)]\!] = \mathsf{unit}_{\multimap} \circ ([\![\mathsf{cut}(y, g)]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} = \mathsf{unit}_{\multimap} \circ (\Lambda_I(\Lambda_I^{-1}[\![y]\!] \circ \Lambda_I^{-1}[\![g]\!]) \multimap \mathsf{id}) \circ \mathsf{lfe} = \mathsf{app}_s \circ (\mathsf{id} \oslash \Lambda_I^{-1} y \circ \Lambda_I^{-1} g)$ by Proposition 2.5.1. This is $\mathsf{app}_s \circ (\mathsf{id} \oslash \Lambda_I^{-1} y) \circ (\mathsf{id} \oslash \Lambda_I^{-1} g) = \mathsf{unit}_{\multimap} \circ (\Lambda_I \Lambda_I^{-1}[\![y]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ (\mathsf{id} \oslash \Lambda_I^{-1} g)$ again by Proposition 2.5.1. This is $\mathsf{unit}_{\multimap} \circ ([\![y]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ (\mathsf{id} \oslash \Lambda_I^{-1} g) = [\![\mathsf{P}^{\lhd}_{\top}(\mathsf{P}^{-}_{\top}(\mathsf{P}_{\lhd}(y)))]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![g]\!]))$ as required.

- If $A = \top$ and $\Gamma = \epsilon$ then $[\![\mathsf{cut}(\mathsf{P}^{+}_{\top}(\mathsf{P}_{\top}), g)]\!] = [\![\mathsf{P}^{+}_{\top}(\mathsf{P}_{\top})]\!] = \mathsf{id}_{\bot} \circ \mathsf{abs} = \mathsf{abs} = \mathsf{abs} \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![g]\!])) = [\![\mathsf{P}^{+}_{\top}(\mathsf{P}_{\top})]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![g]\!]))$ as required.

Finally we show that cut_2 is sound, i.e. $[\![\mathsf{cut}_2(p_1, p_2)]\!] = [\![p_1]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![p_2]\!])$.

- If $Q = Q_1 \oplus Q_2$ then $[\![\mathsf{cut}_2(\mathsf{P}_{\oplus 1}(y), \mathsf{P}_{\&}(a, b))]\!]$
  $= [\![\mathsf{cut}_2(y, a)]\!]$
  $= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![a]\!])$
  $= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \pi_1 \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle)$
  $= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma]\!]^{+}(\pi_1) \circ \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle)$
  $= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma]\!]^{+}(\pi_1)) \circ (\mathsf{id} \otimes \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle)$
  $= [\![y]\!] \circ ([\![\Gamma]\!]^{+}(\pi_1) \oslash \mathsf{id}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle)$
  $= [\![y]\!] \circ [\![\Gamma, N^{\bot}]\!]^{+}(\pi_1) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes (\mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle))$
  $= [\![y]\!] \circ [\![\Gamma, N^{\bot}]\!]^{+}(\pi_1) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}\Lambda_I(\mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle))$
  $= [\![y]\!] \circ [\![\Gamma, N^{\bot}]\!]^{+}(\pi_1) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}((\mathsf{id} \multimap \mathsf{dist}_{+,\Gamma}) \circ \Lambda_I \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle))$
  $= [\![y]\!] \circ [\![\Gamma, N^{\bot}]\!]^{+}(\pi_1) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{dist}_{-,\Gamma^{\bot},P} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2 \rangle \circ \Lambda_I \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle))$
  $= [\![y]\!] \circ [\![\Gamma, N^{\bot}]\!]^{+}(\pi_1) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{dist}_{-,\Gamma^{\bot},P} \circ \langle \Lambda_I \Lambda_I^{-1}[\![a]\!], \Lambda_I \Lambda_I^{-1}[\![b]\!] \rangle))$
  $= [\![y]\!] \circ [\![\Gamma, N^{\bot}]\!]^{+}(\pi_1) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{dist}_{-,\Gamma^{\bot},P} \circ \langle [\![a]\!], [\![b]\!] \rangle))$
  $= [\![\mathsf{P}_{\oplus 1} y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![\mathsf{P}_{\&}(a, b)]\!])$ as required. The case for $\mathsf{P}_{\oplus 2}$ is similar.

- If $Q = Q' \oslash L$ then $[\![\mathsf{cut}_2(\mathsf{P}_{\oslash}(y), \mathsf{P}_{\lhd}(g))]\!] = [\![\mathsf{cut}_2(y, g)]\!] = [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![g]\!]) = [\![\mathsf{P}_{\oslash}(y)]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![\mathsf{P}_{\lhd}(g)]\!])$ as required.

- The case for $Q' \lhd P'$ is similar.

- If $Q = Q_1 \parr Q_2$ then $[\![\mathsf{cut}_2(\mathsf{P}_{\parr 1}(y), \mathsf{P}_{\otimes}(a, b))]\!]$
  $= [\![\mathsf{cut}_2(y, a)]\!]$
  $= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![a]\!])$
  $= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \pi_1 \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!] \rangle)$
  $= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma]\!]^{+}(\pi_1) \circ \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!])$

$= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma]\!]^+(\mathsf{wk} \circ \mathsf{dec}^{-1}) \circ \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!]\rangle)$
$= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma]\!]^+(\mathsf{wk}) \circ [\![\Gamma]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!]\rangle)$
$= [\![y]\!] \circ ([\![\Gamma]\!]^+(\mathsf{wk}) \oslash \mathsf{id}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!]\rangle)$
$= [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\mathsf{wk}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!]\rangle)$
$= [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\mathsf{wk}) \circ \mathsf{wk} \circ \mathsf{sym}(\mathsf{id} \otimes \Lambda_I^{-1}\Lambda_I([\![\Gamma]\!]^+(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!]\rangle))$
$= [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\mathsf{wk}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}((\mathsf{id} \multimap [\![\Gamma]\!]^+(\mathsf{dec}^{-1})) \circ \Lambda_I(\mathsf{dist}_{+,\Gamma} \circ \langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!]\rangle)))$
$= [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\mathsf{wk}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![\Gamma, P^\perp]\!]^+(\mathsf{dec}^{-1}) \circ (\mathsf{id} \multimap \mathsf{dist}_{+,\Gamma}) \circ \Lambda_I\langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!]\rangle))$
$= [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\mathsf{wk}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![\Gamma^\perp, P]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{+,\Gamma,P^\perp} \circ \langle \mathsf{id} \multimap \pi_1, \mathsf{id} \multimap \pi_2\rangle \circ \Lambda_I\langle \Lambda_I^{-1}[\![a]\!], \Lambda_I^{-1}[\![b]\!]\rangle))$
$= [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\mathsf{wk}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![\Gamma^\perp, P]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{-,\Gamma^\perp,P} \circ \langle \Lambda_I\Lambda_I^{-1}[\![a]\!], \Lambda_I\Lambda_I^{-1}[\![b]\!]\rangle))$
$= [\![y]\!] \circ [\![\Gamma, N^\perp]\!]^+(\mathsf{wk}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![\Gamma^\perp, P]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}_{-,\Gamma^\perp,P} \circ \langle [\![a]\!], [\![b]\!]\rangle))$
$= [\![\mathsf{P}_{\parr 1} y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![\mathsf{P}_\otimes(a, b)]\!])$ as required. The case for $\mathsf{P}_{\parr 2}$ is similar.

- If $\Gamma = L, P, \Gamma'$ and $Q = \top$ then $[\![\mathsf{cut}_2(\mathsf{P}_\top^\lhd(y), \mathsf{P}_\bot^\oslash(g))]\!]$
$= [\![\mathsf{cut}(y, g)]\!] = [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![g]\!])$
$= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma']\!]^+(\mathsf{lfe}) \circ [\![\Gamma']\!]^+(\mathsf{lfe}^{-1}) \circ \Lambda_I^{-1}[\![g]\!]))$
$= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma']\!]^+(\mathsf{lfe}) \circ \Lambda_I^{-1}((\mathsf{id} \multimap [\![\Gamma']\!]^+(\mathsf{lfe}^{-1})) \circ [\![g]\!]))$
$= [\![y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma']\!]^+(\mathsf{lfe}) \circ \Lambda_I^{-1}([\![\Gamma', P^\perp]\!]^+(\mathsf{lfe}^{-1}) \circ [\![g]\!]))$
$= [\![y]\!] \circ [\![\Gamma', N^\perp]\!]^+(\mathsf{lfe}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![\Gamma'^\perp, P]\!]^-(\mathsf{lfe}^{-1}) \circ [\![g]\!]))$
$= [\![\mathsf{P}_\top^\lhd y]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![\mathsf{P}_\bot^\oslash g]\!])$ as required.

- The cases for $\Gamma = L, N, \Gamma'$ and $\Gamma = R, \Gamma'$ are similar replacing lfe and lfe^{-1} by the appropriate isomorphism.

- If $A = \top$ and $\Gamma = \epsilon$ then we have $[\![\mathsf{cut}_2(\mathsf{P}_\top^+(\mathsf{P}_\top), \mathsf{P}_\bot^+(y))]\!] = [\![\mathsf{P}_{\parr 2}(\mathsf{wk}_{N^\perp}(y))]\!] = [\![y]\!] \circ \mathsf{unit}_\oslash^{-1} \circ (\mathsf{id} \oslash \mathsf{af}) \circ \mathsf{wk} \circ \mathsf{sym} = [\![y]\!] \circ \mathsf{unit}_\oslash^{-1} \circ \mathsf{wk} \circ (\mathsf{id} \otimes \mathsf{af}) \circ \mathsf{sym} = [\![y]\!] \circ \mathsf{runit}_\otimes^{-1} \circ (\mathsf{id} \otimes \mathsf{af}) \circ \mathsf{sym} = \mathsf{abs} \circ \mathsf{wk} \circ ([\![y]\!] \otimes \mathsf{id}) \circ \mathsf{sym}$ using Proposition 2.5.1. This is $\mathsf{abs} \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![y]\!]) = [\![\mathsf{P}_\top^+(\mathsf{P}_\top)]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![\mathsf{P}_\bot^+(y)]\!]))$ as required.

- If $A = \top$ and $\Gamma = Q$ then we have $[\![\mathsf{cut}_2(\mathsf{P}_\top^\lhd(\mathsf{P}_\top^-(\mathsf{P}_\lhd y)), \mathsf{P}_\bot^\parr(\mathsf{P}_\bot^+(\mathsf{P}_{\parr 1} g)))]\!]$
$= [\![\mathsf{symP}\parr(\mathsf{cut}_2(g, y))]\!]$
$= [\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![y]\!]) \circ \mathsf{sym}$
$= \mathsf{app} \circ (\Lambda([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym}) \otimes \mathsf{id}) \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![y]\!]) \circ \mathsf{sym}$
$= \mathsf{app}_s \circ \mathsf{wk} \circ (\Lambda([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym}) \otimes \Lambda_I^{-1}[\![y]\!]) \circ \circ \mathsf{sym}$
$= \mathsf{app}_s \circ \mathsf{wk} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![y]\!]) \circ (\Lambda([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym}) \otimes \mathsf{id}) \circ \mathsf{sym}$
$= \mathsf{app}_s \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![y]\!]) \circ \mathsf{wk} \circ (\Lambda([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym}) \otimes \mathsf{id}) \circ \mathsf{sym}$
$= \mathsf{app}_s \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![y]\!]) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym}))$

$= \mathsf{unit}_{\multimap} \circ ([\![y]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym}))$ using Proposition 2.5.1

$= \mathsf{unit}_{\multimap} \circ ([\![y]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\Lambda_I \Lambda([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym})))$

$= \mathsf{unit}_{\multimap} \circ ([\![y]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ \Lambda_I([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym})))$

$= \mathsf{unit}_{\multimap} \circ ([\![y]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I([\![g]\!] \circ \mathsf{wk})))$

$= \mathsf{unit}_{\multimap} \circ ([\![y]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ (\Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I([\![g]\!] \circ \mathsf{wk})) \oslash \mathsf{id}) \circ \mathsf{wk} \circ \mathsf{sym}$

$= [\![\mathsf{P}_{\top}^{\lhd}(\mathsf{P}_{\top}^{-}(\mathsf{P}_{\lhd}(y)))]\!] \circ (\Lambda_I^{-1}[\![\mathsf{P}_{\bot}^{\parr}(\mathsf{P}_{\bot}^{+}(\mathsf{P}_{\parr 1}(g)))]\!] \oslash \mathsf{id}) \circ \mathsf{wk} \circ \mathsf{sym}$ as required.

Finally, $[\![\mathsf{cut}_2(\mathsf{P}_{\top}^{\lhd}(\mathsf{P}_{\top}^{-}(\mathsf{P}_{\lhd} y)), \mathsf{P}_{\bot}^{\parr}(\mathsf{P}_{\bot}^{+}(\mathsf{P}_{\parr 2} g)))]\!]$

$= [\![\mathsf{P}_{\parr 2}(\mathsf{cut}(g, y))]\!]$

$= [\![g]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![y]\!]) \circ \mathsf{wk} \circ \mathsf{sym}$

$= [\![g]\!] \circ \mathsf{wk} \circ (\mathsf{id} \otimes \Lambda_I^{-1} y) \circ \mathsf{sym}$

$= \mathsf{app} \circ (\Lambda([\![g]\!] \circ \mathsf{wk}) \otimes \mathsf{id}) \circ (\mathsf{id} \otimes \Lambda_I^{-1} y) \circ \mathsf{sym}$

$= \mathsf{app}_s \circ \mathsf{wk} \circ (\Lambda([\![g]\!] \circ \mathsf{wk}) \otimes \Lambda_I^{-1} y) \circ \mathsf{sym}$

$= \mathsf{app}_s \circ \mathsf{wk} \circ (\mathsf{id} \otimes \Lambda_I^{-1} y) \circ (\Lambda([\![g]\!] \circ \mathsf{wk}) \otimes \mathsf{id}) \circ \mathsf{sym}$

$= \mathsf{app}_s \circ (\mathsf{id} \oslash \Lambda_I^{-1} y) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda([\![g]\!] \circ \mathsf{wk}))$

$= \mathsf{app}_s \circ (\mathsf{id} \oslash \Lambda_I^{-1} y) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ \Lambda_I([\![g]\!] \circ \mathsf{wk})))$

$= \mathsf{app}_s \circ (\mathsf{id} \oslash \Lambda_I^{-1} y) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym})))$

$= \mathsf{unit}_{\multimap} \circ ([\![y]\!] \multimap \mathsf{id}) \circ \mathsf{lfe} \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{pasc}_{\multimap}^{-1} \circ (\mathsf{sym} \multimap \mathsf{id}) \circ \Lambda_I([\![g]\!] \circ \mathsf{wk} \circ \mathsf{sym})))$

$= [\![\mathsf{P}_{\top}^{\lhd}(\mathsf{P}_{\top}^{-}(\mathsf{P}_{\lhd}(y)))]\!] \circ \mathsf{wk} \ \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![\mathsf{P}_{\bot}^{\parr}(\mathsf{P}_{\bot}^{+}(\mathsf{P}_{\parr 2}(g)))]\!]))$ as required. ∎

2.5.3 Isomorphism of Complete WS-categories

We can use the soundness of cut elimination to show that the composition operation in a complete WS-category is in some sense determined, between objects that are the interpretation of some WS formula.

Definition Let $\mathcal{C}$ be a WS-category. We define $\mathcal{C}^{\mathsf{WS}}$ to be a category whose objects are WS-formulas and an arrow $M \to N$ is an element of $\mathcal{C}([\![M]\!], [\![N]\!])$.

Theorem 2.5.6 *If $\mathcal{C}$ and $\mathcal{D}$ are complete WS-categories, then $\mathcal{C}^{\mathsf{WS}}$ and $\mathcal{D}^{\mathcal{WS}}$ are isomorphic.*

Proof Given a proof p of a sequent of WS, we will write $[\![p]\!]_{\mathcal{C}}$ for the denotation of p in the category $\mathcal{C}$.

First, we make an observation. In any complete WS-category, $[\![\mathsf{cut}(\mathsf{reify}(\Lambda_I f), \mathsf{reify}(\Lambda_I g))]\!]$
$= \Lambda_I(\Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I f)]\!]) \circ \Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I g)]\!]))$
$= \Lambda_I(\Lambda_I^{-1}(\Lambda_I f) \circ \Lambda_I^{-1}(\Lambda_I g))$
$= \Lambda_I(f \circ g)$

$= [\![\mathsf{reify}(\Lambda_I(f \circ g))]\!]$. Since $\mathsf{reify}(\Lambda_I(f \circ g))$ and $\mathsf{cut}(\mathsf{reify}(\Lambda_I(f)), \mathsf{reify}(\Lambda_I(g)))$ are two analytic proofs with the same semantics they must be equal, by Theorem 2.4.2.

We define an identity-on-objects functor $F : \mathcal{C}^{\mathsf{WS}} \to \mathcal{D}^{\mathsf{WS}}$. On arrows, we set $F(f) = \Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(f))]\!]_{\mathcal{D}})$.

- F preserves composition: $F(f \circ g)$
  $= \Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(f \circ g))]\!]_{\mathcal{D}})$
  $= \Lambda_I^{-1}([\![\mathsf{cut}(\mathsf{reify}(\Lambda_I f), \mathsf{reify}(\Lambda_I g))]\!]_{\mathcal{D}})$
  $= \Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I f)]\!]_{\mathcal{D}}) \circ \Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I g)]\!]_{\mathcal{D}})$
  $= F(f) \circ F(g)$

- F is surjective on hom-sets: We have $f = F(\Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(f))]\!]_{\mathcal{C}}))$. To see this, note that $F(\Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(f))]\!]_{\mathcal{C}}))$
  $= \Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(\Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(f))]\!]_{\mathcal{C}})))]\!]_{\mathcal{D}})$
  $= \Lambda_I^{-1}([\![\mathsf{reify}([\![\mathsf{reify}(\Lambda_I(f))]\!]_{\mathcal{C}}))]\!]_{\mathcal{D}})$
  $= \Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(f))]\!]_{\mathcal{D}})$
  $= \Lambda_I^{-1}(\Lambda_I(f))$
  $= f$.

- F is injective on hom-sets: If $F(f) = F(g)$ then $\Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(f))]\!]_{\mathcal{D}}) = \Lambda_I^{-1}([\![\mathsf{reify}(\Lambda_I(g))]\!]_{\mathcal{D}})$ so $[\![\mathsf{reify}(\Lambda_I(f))]\!]_{\mathcal{D}} = [\![\mathsf{reify}(\Lambda_I(g))]\!]_{\mathcal{D}}$. Then by applying reify to both sides, we see that $\mathsf{reify}(\Lambda_I(f)) = \mathsf{reify}(\Lambda_I(g))$. By taking the semantics of both sides in $\mathcal{C}$ we see that $\Lambda_I(f) = \Lambda_I(g)$, and so $f = g$.

- F preserves the identity: We can show that $F(\mathsf{id})$ is the identity in $\mathcal{D}^{\mathsf{WS}}$ using the above facts. In particular $F(\mathsf{id}) \circ f = F(\mathsf{id}) \circ F(F^{-1}(f)) = F(\mathsf{id} \circ F^{-1}(f)) = F(F^{-1}(f)) = f$ and $f \circ F(\mathsf{id}) = F(F^{-1}(f)) \circ F(\mathsf{id}) = F(F^{-1}(f) \circ \mathsf{id}) = F(F^{-1}(f)) = f$.

- F is surjective on objects: This is clear, as F is identity-on-objects. ∎

2.6 Embedding Polarized Linear Logic in WS

Polarized Linear Logic [53] is a polarisation of linear logic into negative and positive formulas, representing computation and value types, broadly speaking. The linear fragment MALLP can be embedded in WS, as families of formulas and proofs. We describe the embedding here, which will be of use in Chapter 5.

In the proof derivations in the rest of this thesis, we will not always label core rules when the label can be inferred from the premises and the conclusion. Further, we will

Figure 2-14: Proof rules for MALLP

$$ax\ \frac{}{\vdash N, N^\perp} \qquad cut\ \frac{\vdash \Gamma, N \qquad \vdash \Delta, N^\perp}{\vdash \Gamma, \Delta} \qquad \otimes\ \frac{\vdash \Gamma, P \qquad \vdash \Delta, Q}{\vdash \Gamma, \Delta, P \otimes Q}$$

$$\parr\ \frac{\vdash \Gamma, M, N}{\vdash \Gamma, M \parr N} \qquad \oplus_1\ \frac{\vdash \Gamma, P}{\vdash \Gamma, P \oplus Q} \qquad \oplus_2\ \frac{\vdash \Gamma, Q}{\vdash \Gamma, P \oplus Q}$$

$$\&\ \frac{\vdash \Gamma, M \qquad \vdash \Gamma, N}{\vdash \Gamma, M \& N} \qquad \mathbf{1}\ \frac{}{\vdash \mathbf{1}} \qquad \bot\ \frac{\vdash \Gamma}{\vdash \Gamma, \bot}$$

$$\top\ \frac{}{\vdash \Gamma', \top} \qquad \downarrow\ \frac{\vdash \Gamma^-, N}{\vdash \Gamma^-, \downarrow N} \qquad \uparrow\ \frac{\vdash \Gamma, P}{\vdash \Gamma, \uparrow P}$$

$$ex\ \frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta}$$

often combine sequences of core rules into a single rule, when the sequence is unique and can be inferred (in particular the core elimination rules and $\mathsf{P}_\oslash$,$\mathsf{P}_\triangleleft$).

The formulas of MALLP are as follows:

$$\begin{array}{rcccccccccc} P & := & \mathbf{1} & | & \mathbf{0} & | & P \otimes Q & | & P \oplus Q & | & \downarrow N \\ N & := & \bot & | & \top & | & M \parr N & | & M \& N & | & \uparrow P \end{array}$$

There is an operation $(-)^\perp$ exchanging polarity, swapping $\mathbf{1}$ for $\bot$, $\mathbf{0}$ for $\top$, $\otimes$ for $\parr$, and so on.

We can immediately note the similarity between the connectives of LLP and some of the connectives of WS!, although the polarities do not match up (see Section 2.2.1).

A sequent of MALLP is a list of MALLP formulas. The proof rules for Polarized Linear Logic are given in Figure 2-14. Γ^- ranges over lists of negative formulas, and Γ' over lists where at most one formula is positive. Each provable sequent has at most one positive formula, so we can restrict our attention to sequents of this form. It is possible to give semantics to MALLP proofs as *innocent* strategies [53], which do not have access to the entire history of play.

We next describe an embedding of MALLP inside WS. Apart from some renaming of units, connectives in LLP will be interpreted by the connective of the same name in WS. Broadly speaking, positive formulas of LLP will be mapped to negative formulas of WS, and negative formulas of LLP to positive formulas of WS. However, under this scheme there is a mismatch for the additives: we will resultantly need to map formulas of LLP to *families* of WS formulas. For example, we can represent a "disjunction" of negative formulas by a family of negative formulas together with a proof of one of them. The formulas that have a lift as their outermost connective will be mapped to singleton families.

Figure 2-15: MALLP formulas as families of WS formulas

$A \in \mathsf{LLP}$	$i(A) \in \mathsf{Fam\ WS}$
$\mathbf{1}$	$(\{*\}, \_ \mapsto \mathbf{1})$
$\mathbf{0}$	$(\{*\}, \_ \mapsto \bot)$
$P \otimes Q$	$(\lvert i(P)\rvert \times \lvert i(Q)\rvert, \langle x, y\rangle \mapsto i(P)_x \otimes i(Q)_y)$
$P \oplus Q$	$(\lvert i(P)\rvert \uplus \lvert i(Q)\rvert, [\mathsf{in}_1(x) \mapsto i(P)_x, \mathsf{in}_2(y) \mapsto i(Q)_y]$
$\downarrow N$	$(\{*\}, \_ \mapsto \&_{j \in \lvert i(N)\rvert}(\bot \lhd i(N)_j)$
$\bot$	$(\{*\}, \_ \mapsto \mathbf{0})$
$\top$	$(\{*\}, \_ \mapsto \top)$
$M \parr N$	$(\lvert i(M)\rvert \times \lvert i(N)\rvert, \langle x, y\rangle \mapsto i(M)_x \parr i(N)_y))$
$M \& N$	$(\lvert i(M)\rvert \uplus \lvert i(N)\rvert, [\mathsf{in}_1(x) \mapsto i(M)_x, \mathsf{in}_2(y) \mapsto i(N)_y]$
$\uparrow P$	$(\{*\}, \_ \mapsto \bigoplus_{j \in \lvert i(P)\rvert}(\top \oslash i(P)_j)$

2.6.1 Translation of Formulas

Let WS^- denote the set of negative WS formulas, and WS^+ the set of positive WS formulas.

Definition A *finite family of negative (resp. positive) WS formulas* is a pair (I, f) where I is a finite set and $f : I \to \mathsf{WS}^-$ (resp. $I \to \mathsf{WS}^+$).

For brevity, given such a family $F = (I, f)$ we will write $|F|$ for I and F_x for $f(x)$.

We will interpret a negative formula of MALLP as a finite family of positive WS formulas, and a positive formula of MALLP as a finite family of negative WS formulas. We describe this mapping in Figure 2-15. We can see that $|i(A^\perp)| = |i(A)|$ and $i(A^\perp)_y = i(A)_y^\perp$.

By composing this mapping with our interpretation of WS formulas, one obtains a semantics of LLP formulas with respect to families of games, inverting polarity. This differs from the original innocent games model of LLP [53], which does not invert polarity. In Laurent's setting, the interpretation of tensor is a non-standard operation on positive games, where Player starts and Opponent may switch. This is facilitated by the fact that Player's first move in $P \otimes Q$ is a pair of moves (p, q) from each component. We could give an embedding of LLP into WS based on this semantics by defining Laurent's positive $\otimes$ operator as a macro on positive WS formulas and appealing to full completeness, but we instead chose this embedding to use the non-core rules of WS directly.

2.6.2 Translation of Proofs

We translate proofs of LLP to families of proofs of WS in the following manner:

- Given an LLP proof p of $\vdash N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$ for each i, we construct a proof $i(p, \overrightarrow{x_i})$ of $\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$
- Given an LLP proof p of $\vdash N_1, \ldots, N_i, Q, N_{i+1}, \ldots, N_n$ and $x_i \in |i(N_i)|$ for each i, we construct a pair $i(p, \overrightarrow{x_i}) = (y, q)$ where $y \in |i(Q)|$ and q is a proof of $\vdash i(Q)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$.

Proposition 2.6.1 *For each LLP formula P, $y \in |i(P)|$ and sequence of negative WS formulas Δ^- there is a WS proof $\mathsf{P}^\top_{P,y} \vdash i(P)_y, \Delta^-, \top$*

Proof Simple induction on P. Set $\mathsf{P}^\top_{\mathbf{1},*} = \mathsf{P}_{\mathbf{1}}$ and $\mathsf{P}^\top_{\mathbf{0},*} =$

$$(\mathsf{P}^-_\bot)^* \dfrac{\dfrac{\overline{\vdash \top}}{\vdash \bot, \top}}{\vdash \bot, \Delta^-, \top}$$

where $(\mathsf{P}^-_\bot)^*$ denotes repeated use of $\mathsf{P}^-_\bot$. Let $\mathsf{P}^\top_{P_1 \oplus P_2, \mathsf{in}_i(q)} = \mathsf{P}^\top_{P_i,q}$ and $\mathsf{P}^\top_{P_1 \otimes P_2, (y_1,y_2)} = \mathsf{P}_\otimes(\mathsf{P}^\top_{P_1,y_1}, \mathsf{P}^\top_{P_2,y_2})$. $\mathsf{P}^\top_{\downarrow N,*}$ is defined as follows:

$$\dfrac{\begin{array}{c}(\mathsf{P}^-_\bot)^* \dfrac{\dfrac{\vdots}{\vdash \top}}{\vdash \bot, \top} \\ \mathsf{P}^+_{\mathsf{wk}} \dfrac{\vdots}{\vdash \bot, \Delta^-, \top} \\ \dfrac{\vdots}{\vdash \bot, i(N)_j, \Delta^-, \top} \\ \dfrac{\vdots}{\vdash \bot \lhd i(N)_j, \Delta^-, \top}\end{array} \quad \vdots}{\vdash \&_{j \in |i(N)|}(\bot \lhd i(N)_j), \Delta^-, \top}$$

with each branch following the same shape. ∎

We next show how each of the MALLP proof rules are translated.

- The *ax* rule, with $p = ax$: For each $x \in |i(N)|$ we set $i(p, x) = (y, q)$ where $y = x \in |i(N^\bot)|$ and q is the proof of $\vdash i(N^\bot)_x, i(N)_x$ given by P_{id}.
- The *cut* rule, with $p = cut(q, r)$:

  Suppose $\Gamma = N_1, \ldots, N_n$ and $\Delta = M_1, \ldots, M_m$. Let $x_i \in |i(N_i)|$ and $y_i \in |i(M_i)|$. Then $i(r, \overrightarrow{y_i}) = (y, t)$ with $y \in |i(N^\bot)|$ and $t \vdash i(N^\bot)_y, i(M_1)_{y_1}, \ldots, i(M_n)_{y_n}$. Then $i(q, \overrightarrow{x_i}, y) \vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_y$. Applying $\mathsf{P}_{\mathsf{cut}}$ to this proof and t results in a proof g of $\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M_1)_{y_1}, \ldots, i(M_m)_{y_m}$ and we set $i(p, \overrightarrow{x_i}, \overrightarrow{y_i}) = g$.

  Suppose $\Gamma = N_1, \ldots, N_i, P, N_{i+1}, \ldots N_n$ and $\Delta = M_1, \ldots, M_m$. Let $x_i \in |i(N_i)|$ and $y_i \in |i(M_i)|$. Then $i(r, \overrightarrow{y_i}) = (y, t)$ with $y \in |i(N^\bot)|$ and $t \vdash i(N^\bot)_y, i(M_1)_{y_1}, \ldots, i(M_n)_{y_n}$.

Then $i(q, \overrightarrow{x_i}, y) = (y', q')$ where $y' \in |i(P)|$ and $q' \vdash i(P)_{y'}, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_y$. Applying $\mathsf{P}_{\mathsf{cut}}$ to this proof and t results in a proof g of

$$\vdash i(P)_{y'}, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M_1)_{y_1}, \ldots, i(M_m)_{y_m}$$

and we set $i(p, \overrightarrow{x_i}, \overrightarrow{y_i}) = (y', g)$.

- The $\otimes$ rule, with $p = \otimes(q_1, q_2)$: Suppose that $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$ and $\Delta = M_1, \ldots M_m$ with $y_i \in |i(M_i)|$. Let $i(q_1, \overrightarrow{x_i}) = (w, q_1')$ with $w \in |i(P)|$ and $q_1' \vdash i(P)_w, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$. Let $i(q_2, \overrightarrow{y_i}) = (z, q_2')$ with $q_2' \vdash i(Q)_z, i(M_1)_{y_1}, \ldots, i(M_m)_{y_m}$. By applying $\mathsf{P}_{\mathsf{mul}}$ and $\mathsf{P}_{\otimes}$, we construct a proof

$$pq \vdash i(P)_w \otimes i(Q)_z, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M_1)_{y_1}, \ldots, i(M_m)_{y_m}$$

We set $i(p, \overrightarrow{x_i}, \overrightarrow{y_i}) = ((w, z), pq)$.

- The $\parr$ rule, with $p = \parr(q_1, q_2)$:

  Suppose that $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$ and $x \in |i(M \parr L)|$. Then $x = (m, l)$ with $m \in |i(M)|$ and $l \in |i(L)|$. Then $i(q, \overrightarrow{x_i}, m, l) = q'$ where $q' \vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M)_m, i(L)_l$. Let $q'' = \mathsf{P}^{\mathsf{T}}_{\parr}(q)$, a proof of

$$\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M)_m \parr i(L)_l$$

  We set $i(p, \overrightarrow{x_i}, x) = q''$.

  Suppose that $\Gamma = N_1, \ldots, N_i, P, N_{i+1}, \ldots, N_n$ and $x_i \in |i(N_i)|$ and $x \in |i(M \parr L)|$. Then $x = (m, l)$ with $m \in |i(M)|$ and $l \in |i(L)|$. Then $i(q, \overrightarrow{x_i}, m, l) = (y, q')$ where $y \in |i(P)|$ and $q' \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M)_m, i(L)_l$. Let $q'' = \mathsf{P}^{\mathsf{T}}_{\parr}(q)$, a proof of $\vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M)_m \parr i(L)_l$. We set $i(p, \overrightarrow{x_i}, x) = (y, q'')$.

- The $\oplus 1$ rule, with $p = \oplus 1(q)$: Suppose that $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$. Then $i(q, \overrightarrow{x_i}) = (y, q')$ where $y \in |i(P)|$ and $q' \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$. We set $i(p, \overrightarrow{x_i}) = (\mathsf{in}_1(y), q')$. The case for $\oplus 2$ is similar.

- The & rule, proving $\vdash \Gamma, M_1 \& M_2$ with $p = \&(q_1, q_2)$:

  Suppose that $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$ and $x \in |i(M_1 \& M_2)|$. Then $x = \mathsf{in}_j(x')$ with $x' \in |i(M_j)|$. Then $i(q_j, \overrightarrow{x_i}, x') = q'$ with $q' \vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M_j)_{x'}$. Set $i(p, \overrightarrow{x_i}, x) = q'$.

  Suppose that $\Gamma = N_1, \ldots, N_i, P, N_{i+1}, \ldots, N_n$ and $x_i \in |i(N_i)|$ and $x \in |i(M_1 \& M_2)|$. Then $x = \mathsf{in}_j(x')$ with $x' \in |i(M_j)|$. Then $i(q_j, \overrightarrow{x_i}, x') = (y, q')$ with $y \in |i(P)|$ and $q' \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(M_j)_{x'}$. Set $i(p, \overrightarrow{x_i}, x) = (y, q')$.

- The $\mathbf{1}$ rule: We set $i(\mathbf{1}) = (*, \mathsf{P}_{\mathbf{1}})$.

- The $\bot$ rule, with $p = \bot(q)$:

  Suppose that $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$. Then $i(q, \overrightarrow{x_i}) = q'$ which is a WS proof of $\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$. By applying $\mathsf{P}_0^{\mathsf{T}}$ this yields a proof $q'' \vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, \mathbf{0}$. We thus set $i(p) = q''$.

  Suppose that $\Gamma = N_1, \ldots, N_i, P, N_{i+1}, \ldots N_n$ and $x_i \in |i(N_i)|$. Then $i(q, \overrightarrow{x_i}) = (y, q')$ with $y \in |i(P)|$ and $q' \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$. By applying $\mathsf{P}_0^{\mathsf{T}}$ this yields a proof $q'' \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, \mathbf{0}$. We thus set $i(p) = (y, q'')$.

- The $\top$ rule:

  Suppose $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$. We must give a proof of $\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, \top$ and we use $\mathsf{P}^{\top}_{\mathbf{0},*}$ of Proposition 2.6.1.

  Suppose $\Gamma = N_1, \ldots, N_i, P, N_{i+1}, \ldots N_n$ and $x_i \in |i(N_i)|$. We must give a $y \in |i(P)|$ and a proof of $\vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, \top$. Since $|i(P)|$ is finite, we can chose an arbitrary element, and give the proof of $\vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, \top$ using $\mathsf{P}^{\top}_{P,y}$ of Proposition 2.6.1.

- The $\downarrow$ rule: Let $\Gamma^- = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$. For each $j \in |i(N)|$, $i(q, \overrightarrow{x_i}, j) \vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_j$. We then perform the following derivation r_j:

$$\mathsf{P}_{\mathsf{sym}} \dfrac{\dfrac{\overline{i(q, \overrightarrow{x_i}, n) \vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_j}}{\vdash \bot, i(N)_j, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}{\vdash \bot \lhd i(N)_j, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}$$

  Using many applications of $\mathsf{P}_{\&}$ we can construct a proof

$$r \vdash \&_{j \in |i(N)|}(\bot \lhd i(N)_j), i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$$

  We then set $i(p, \overrightarrow{x_i}) = (*, r)$.

- The $\uparrow$ rule, with $p = \uparrow(q)$: Let $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$. Then $i(q, \overrightarrow{x_i}) = (y, q)$ where $q \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$. We set $i(p, \overrightarrow{x_i})$ to be the following proof:

$$\dfrac{\mathsf{P}_{\oplus y} \dfrac{\dfrac{\dfrac{q \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}{\vdash \top, i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}{\vdash \top \oslash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}{\vdash \bigoplus_{j \in |i(P)|} \top \oslash i(P)_j, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}{\vdash \bot, i(N_1)_{x_1} \parr \ldots \parr i(N_n)_{x_n} \parr (\bigoplus_{j \in |i(P)|} \top \oslash i(P)_j)}}{\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, \bigoplus_{j \in |i(P)|} \top \oslash i(P)_j}$$

Note that in the semantics of this rule two moves are played: the opening lift overall (O-move) and the the opening lift in the derelicted component (P-move), which corresponds to "focusing" on that component.

- The ex rule, with $p = ex(q)$:

  If either A or B are positive, then we can set $i(p) = i(q)$.

  Next suppose A and B are both negative, $\Gamma = N_1, \ldots, N_n$ and $\Delta = M_1, \ldots, M_m$. Let $x_i \in |i(N_i)|$; $y_i \in |i(M_i)|$; $a \in |i(A)|$ and $b \in |i(B)|$. Then $i(q, \overrightarrow{x_i}, a, b, \overrightarrow{y_i}) \vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(A)_a, i(B)_b, i(M_1)_{y_1}, , i(M_m)_{y_m}$ and by applying $\mathsf{P_{sym}}$ we obtain a proof q' of

  $$\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(B)_b, i(A)_a, i(M_1)_{y_1}, \ldots, i(M_m)_{y_m}$$

  and set $i(p, \overrightarrow{x_i}, b, a, \overrightarrow{y_i}) = q'$.

  Next suppose A and B are both negative, with $\Gamma = N_1, \ldots, N_n$ and $\Delta = M_1, \ldots, M_i, P, M_{i+1}, \ldots M_m$. Let $x_i \in |i(N_i)|$; $y_i \in |i(M_i)|$; $a \in |i(A)|$ and $b \in |i(B)|$. Then $i(q, \overrightarrow{x_i}, a, b, \overrightarrow{y_i}) = (y, q')$ where $y \in |i(P)|$ and

  $$q \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(A)_a, i(B)_b, i(M_1)_{y_1}, \ldots, i(M_m)_{y_m}$$

  and by applying $\mathsf{P_{sym}}$ we obtain a proof q' of

  $$\vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(B)_b, i(A)_a, i(M_1)_{y_1}, \ldots, i(M_m)_{y_m}$$

  and set $i(p, \overrightarrow{x_i}, b, a, \overrightarrow{y_i}) = q'$.

  The case where $\Gamma = N_1, \ldots, N_i, P, N_{i+1}, \ldots, N_n$ is entirely similar.

We can hence interpret proofs in MALLP as (families of) proofs in WS.

This concludes our treatment of the multiplicative-additive kernel of WS. In the next chapters, we will see how further expressivity can be added. First, we will consider an exponential structure that leads us into the realm of infinite games.

Chapter 3

Exponentials

The logic introduced in Chapter 2 is affine — a proof of $A \multimap B$ may use its argument at most once. We next introduce exponentials into WS, yielding the full expressivity of Intuitionistic Linear Logic. This leads us to consider infinite games and winning conditions. In this setting we represent the exponential in the logic explicitly as a final coalgebra.

3.1 Introduction

3.1.1 Exponentials in Game Semantics

We next introduce exponential modalities into our logic. This is a unary operator ! that transforms a resource A that can only be used once to a reusable resource $!A$: there are maps $!A \multimap !A \otimes !A$ and $!A \multimap A$. Since proofs correspond to games, it is in the spirit of our programme to explicitly base the exponential on some exponential comonad in the category of games. As identified in [58], there are multiple choices:

- A non-repetitive backtracking exponential introduced by Lamarche [52]
- An exponential based on infinitely many copies of the subgame [7,36]
- A backtracking exponential where repetition is allowed [31].

We will choose to study the second exponential, for its use in modelling imperative programming languages. For example, the game model of Idealized Algol given in [8] is set in the Kleisli category of this comonad. Thus, we will be able to embed Idealized Algol over finitary ground types in our logic. The other exponentials also have uses in modelling languages: for example, a fully abstract model of PCF with `catch` can be given using the first exponential [22] and the third can be used to formalise innocent strategies over an arena [31], an interpretation of purely functional programs.

3.1.2 Chapter Overview

In this chapter, we will first describe this exponential on games. We will observe that this second exponential $!A$ is the carrier of the *final coalgebra* of the functor $X \mapsto A \oslash X$ and show how this fact can be used to derive both the usual linear exponential structure (promotion) and also stateful behaviour: for example, a reusable Boolean cell is the anamorphism of a finite strategy.

We can represent the fact that $!A$ is the final coalgebra of this functor in the logic, in the style of [20]. Using the above observations, we can then encode both full Intuitionistic Linear Logic and stateful objects such as a Boolean cell. We give semantics to the resulting logic.

We will then consider full completeness. The reification procedure from Chapter 2 extends in a simple manner, but it only terminates for finite strategies (without the exponentials, all strategies are finite). However, we will see that we can reify arbitrary total strategies as *infinitary* analytic proofs — analytic proofs that may have infinite depth — and so for each proof, we can construct the unique infinitary analytic proof with the same semantics.

3.1.3 The Need for a Winning Condition

Our proofs are interpreted as total strategies on the appropriate game. However, total strategies only compose when the underlying games are bounded. For example, we note that there are unique total strategies $\sigma : I \to \Sigma^*$ and $\tau : \Sigma^* \multimap \bot$ where Σ^* is as in Section 2.1.4. However the composition $\tau \circ \sigma : I \to \bot$ is the empty strategy, which is not total. The source of the problem is that 'infinite chattering' can occur in the hidden component, so the three-way dialogue continues to be live but no moves are externally visible.

I	$\to$	Σ^*	$\to$	$\bot$	
				q	O
		q			P
		a			O
		q			P
		a			O
		$\vdots$			

For the semantics of WS, this is not a problem as the denotation of every formula is bounded. However, our exponential of choice does not preserve boundedness.

A standard solution to this problem is to introduce winning conditions on games:

we add an extra component to the definition of game describing which infinite plays are won by Player and which are won by Opponent. We follow the approach of [36]. In particular, an infinite play on $A \multimap B$ is P-winning if it is P-winning on B or O-winning on A. We then require that strategies contain only winning plays. This breaks the example above, as the unique infinite play in Σ^* must be designated either P-winning or O-winning. In the former case τ is not a winning strategy; in the latter case σ is not a winning strategy.

3.2 Games and Winning Conditions

3.2.1 Win-games

If A is a set, let A^ω denote the set of infinite sequences on A; and if s is such a sequence write $t \sqsubset s$ if t is a finite prefix of s. We extend the restriction notation in Section 2.1.2 to infinite plays, noting that $s|_i$ may be finite or infinite. Dually, if $s \in X_i^\omega$ we write $\mathsf{in}_i^\omega(s)$ for the corresponding sequence in $(X_1 + X_2)^\omega$.

If A is a game, let $\overline{P_A} = \{s \in M_A^\omega : \forall t \sqsubset s.t \in P_A\}$. Thus $\overline{P_A}$ represents the infinite limits of plays in P_A (the 'infinite plays'). We add a notion of winning conditions to games, as in [36].

Definition A *win-game* is a tuple $(M_A, \lambda_A, b_A, P_A, W_A)$ where $(M_A, \lambda_A, b_A, P_A)$ is a game and $W_A \subseteq \overline{P_A}$.

W_A represents the set of infinite plays that are P-winning; we say an infinite play is O-winning if it is not P-winning. We can extend the notion of winner of a play to finite plays, where the last player to play a move in that play wins. More formally, let we let $W_A^* = W_A \cup E_A$ where E_A is the set of plays that end in a P-move.

3.2.2 Connectives on Win-games

We extend our connectives on games to connectives on win-games.

- Units — For $A \in \{\mathbf{1}, \mathbf{0}, \top, \bot\}$, $\overline{P_A} = \emptyset$ — there are no infinite plays, and so we take $W_A = \emptyset$, as we must.
- Tensor — An infinite play in $M \otimes N$ is P-winning if its restriction to M is P-winning and its restriction to N is P-winning. Formally, $W_{M \otimes N} = \{s \in P_{M \otimes N}^\omega : s|_1 \in W_M^* \wedge s|_2 \in W_N^*\}$.
- Sequoid — Similarly, an infinite play in $A \oslash N$ is P-winning if both of its restrictions are P-winning: $W_{A \oslash N} = \{s \in P_{A \oslash N}^\omega : s|_1 \in W_A^* \wedge s|_2 \in W_N^*\}$.

- Par — An infinite play in $P \parr Q$ is P-winning if its restriction to P is P-winning or its restriction to Q is P-winning. Formally, $W_{P \parr Q} = \{s \in P^{\omega}_{P \parr Q} : s|_1 \in W^*_P \vee s|_2 \in W^*_Q\}$.

- Cosequoid — Similarly, an infinite play in A play in $A \lhd P$ is P-winning if either of its restrictions are P-winning: $W_{A \lhd Q} = \{s \in P^{\omega}_{A \lhd Q} : s|_1 \in W^*_A \vee s|_2 \in W^*_Q\}$.

- Product — An infinite play in $M \& N$ is P-winning if its restriction to M is P-winning (if it is a play in M) or its restriction to N is P-winning (if it is a play in N). Thus, $W_{M \& N} = \{\mathsf{in}^{\omega}_1(s) : s \in W_M\} \cup \{\mathsf{in}^{\omega}_2(s) : s \in W_N\}$.

- Sum — Similarly, an infinite play in $P \oplus Q$ is P-winning if it is P-winning in the relevant component: $W_{P \oplus Q} = \{\mathsf{in}^{\omega}_1(s) : s \in W_P\} \cup \{\mathsf{in}^{\omega}_2(s) : s \in W_Q\}$.

We extend negation $A^{\perp}$ to win-games by setting $W_{A^{\perp}} = \overline{P_A} - W_A$. We have the usual duality between $\otimes$ and $\parr$, $\oslash$ and $\lhd$, & and $\oplus$.

Again we set $M \multimap N = N \lhd M^{\perp}$. Then a play $s \in \overline{P_{M \multimap N}}$ is P-winning if $s|_N$ is P-winning or $s|_{M^{\perp}}$ is P-winning, i.e. if $s|_N$ is P-winning or $s|_M$ is O-winning, i.e. if $s|_M$ P-winning implies $s|_N$ P-winning.

3.2.3 Winning Strategies

Our notion of (total) strategies on games lifts to win-games.

Definition A strategy σ on a win-game A is *winning* if

- σ is total

- If $s \in \overline{P_A}$ and all prefixes of s ending in a P-move are in σ, then $s \in W_A$.

It is known that the composition of two winning strategies is itself winning [36].

Proposition 3.2.1 *If $\sigma : M \multimap N$ and $\tau : N \multimap L$ are winning strategies then $\tau \circ \sigma$ is winning.*

Proof If $\tau \circ \sigma$ is not total then there must be an infinite interaction sequence s with all finite prefixes in $\sigma \| \tau$ with infinite chattering in N: $s = tr$ where all moves in r occur in N. Suppose $s|_N = t|_N r \in W_N$. Then since τ is winning and $t|_{N,L} r \in \tau$ it follows that $t|_L \in W^*_L$. Since this play is finite, it must end in a Player-move. But this is impossible since in any play in $M \multimap N$ it must be Player who switches components. So we must have $s|_N \notin W_N$. But then since σ is winning we must have $t|_M \notin W^*_M$ i.e. $t|_M$ ends in an O-move. But this means that $t|_{M \multimap N}|_M$ ends in a P-move, which is again impossible

by the switching condition for σ. Thus such infinite chattering cannot occur, and $\tau \circ \sigma$ is total.

We can also show that $\tau \circ \sigma$ satisfies the second condition. Let $t \in P^{\omega}_{M \multimap L}$ be such that each prefix of t ending in a P-move is in $\tau \circ \sigma$. Then there is a unique interaction sequence s such that $s|_{M,L} = t$ and each finite prefix of s is in $\sigma \| \tau$. We need to show that $s|_{M,L} \in W_{M \multimap L}$ i.e. $s|_M \in W^*_M \Rightarrow s|_L \in W^*_L$. Suppose that $s|_M \in W^*_M$. Then since σ is winning and all finite prefixes of $s|_{M,N}$ are in σ, we must have $s|_{M,N} \in W_{M \multimap N}$. But since $s|_M \in W^*_M$ we must have $s|_N \in W^*_N$. Then since τ is winning and all finite prefixes of $s|_{N,L}$ are in τ we must have $s|_L \in W^*_L$. Thus, $t = s|_{M,L} \in W_{M \multimap L}$, as required. ∎

3.2.4 Categorical Structure

We know that winning strategies compose, and it is easy to see that the identity strategy $\mathsf{id}_A : A \multimap A$ is winning for any A. We can thus construct a category $\mathcal{W}$ of win-games and winning strategies. This category is symmetric monoidal closed with respect to $(I, \otimes, \multimap)$ and $(I, \&)$ provides finite products [36]. Once again we can identify a sequoidal closed structure with an action $\oslash : \mathcal{W}_s \times \mathcal{W} \to \mathcal{W}_s$ where $\mathcal{W}_s$ is the full-on-objects subcategory of $\mathcal{W}$ given by strategies that are both strict and winning. The mediating isomorphisms are all essentially copycat strategies, as in $\mathcal{G}$.

Proposition 3.2.2 *$\mathcal{W}$ is a complete WS-category.*

We can thus interpret WS inside $\mathcal{W}$ in a fully complete manner.

There is a functor $U : \mathcal{W} \to \mathcal{G}$ that simply forgets the winning condition. There is also a functor $F : \mathcal{G}_t \to \mathcal{W}$ with $F(A) = (M_A, \lambda_A, P_A, \emptyset)$. Note that $U(F(A)) = A$ and if A is bounded then $F(U(A)) = A$.

We will write $\mathcal{G}_w$ for the category of win-games and (not-necessarily winning) strategies. Since $\mathcal{G}_w$ is equivalent to $\mathcal{G}$, we will often omit the subscript $_w$ if its presence is irrelevant or can be inferred.

3.3 Sequoidal Exponential as a Final Coalgebra

We now describe the exponential comonad on $\mathcal{W}$ that we have chosen to study. This appears in [7,36].

3.3.1 Sequoidal Exponential

Exponential Connectives

Let N be a negative win-game. We can consider $!N$ to be the game consisting of an infinite number of copies of N, where Opponent can spawn new copies and switch between copies he has opened. The game $!N$ is played over $M_N \times \mathbb{N}$ and copies must be opened in successive order. An infinite play is winning just if it is winning in each component. Thus, we define

$$!N = (M_N \times \mathbb{N}, \lambda_N \circ \pi_1, \{s : \forall i.s|_i \in P_N \wedge s|_i = \epsilon \Rightarrow s|_{i+1} = \epsilon\}, \{s : \forall i.s|_i \in W_N^*\}).$$

As with the tensor, there is an implicit switching condition: since the play overall must be alternating, and the play restricted to each component must be alternating, it follows that only Opponent can switch between copies and open new copies.

Dually, if Q is a positive game we define $?Q$ to be the game consisting of an infinite number of copies of Q, where Player can spawn new copies and switch between them. An infinite play is winning if it is winning in at least one component.

$$?Q = (M_Q \times \mathbb{N}, \lambda_Q \circ \pi_1, \{s : \forall i.s|_i \in P_Q \wedge s|_i = \epsilon \Rightarrow s|_{i+1} = \epsilon\}, \{s : \exists i.s|_i \in W_Q^*\})$$

Given a strategy $\sigma : N$, we can construct a strategy $\sigma^\dagger : !N$ which, in each copy, behaves as σ. However, since our strategies are history-sensitive, there are also strategies on $!N$ which behave *differently* in each copy, and whose behaviour in one copy can depend on what has previously happened in other copies.

Linear Exponential Comonad

It is well known that we can model the exponential modality of linear logic in a symmetric monoidal closed category by a *linear exponential comonad* [72]. This is a functor $! : \mathcal{C} \to \mathcal{C}$ with maps $\mathsf{der} : !A \to A$, $\mathsf{mult} : !A \multimap !!A$ and $\mathsf{d} : !A \multimap !A \otimes !A$ satisfying some further structure and properties. There is a promotion operator $(-)^\dagger$ taking a map $!A \to B$ to a map $!A \to !B$. Each of the exponentials defined in Section 3.1.1 are linear exponential comonads.

For our chosen exponential, the der, mult and d operators are all copycat strategies.

- The strategy $\mathsf{der} : !A \multimap A$ simply uses the first copy of A in $!A$.
- In $\mathsf{mult} : !A \to !!A$, each time a copy of A is opened on the right in $!!A$ (whether it is an 'inner' copy or an 'outer' copy) a new copy of A is opened on the left.

- For $\mathsf{d} : !A \multimap !A \otimes !A$, each new copy of A in $!A \otimes !A$ (whether it is in the left component or the right component) opens a new copy of A in the argument.
- If $\sigma : !A \to B$ the strategy $\sigma^\dagger : !A \to !B$ behaves as σ in each copy.

For any linear exponential comonad, we have isomorphisms $!(A \& B) \cong !A \otimes !B$: again in our current setting these are just copycat strategies.

Cofree Commutative Comonoid

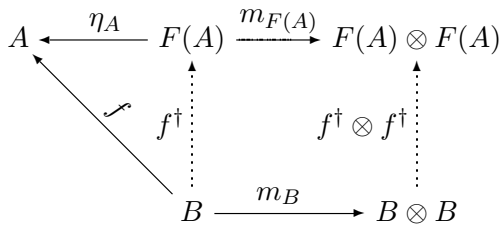
It is also known that a *cofree commutative comonoid* [42, 62] is sufficient to model the linear logic exponential in a symmetric monoidal category. This also provides the operations described above, derived from a stronger universal property.

A *comonoid* in a category $\mathcal{C}$ is an object A together with maps $m : A \multimap A \otimes A$ and $u : A \to I$ satisfying some coherence conditions (associativity and identity). In our setting since I is the terminal object, $u = \mathsf{t}$ and we need not consider it as part of the data for a comonoid. A morphism of comonoids $f : (A, m_A) \to (B, m_B)$ is a morphism $f : A \to B$ such that $m_B \circ f = (f \otimes f) \circ m_A$.

A comonoid is *commutative* if $\mathsf{sym} \circ m = m$. In a decomposable sequoidal category there is an isomorphism $\mathsf{dec} : A \otimes A \cong (A \oslash A) \times (A \oslash A)$, and the symmetry law is equivalent to $\pi_1 \circ \mathsf{dec} \circ m = \pi_2 \circ \mathsf{dec} \circ m$, i.e. $\mathsf{dec} \circ m = \langle d, d \rangle$ for some map $d : A \to A \oslash A$. Thus, in a sequoidal closed category, a commutative comonoid is equivalent to a map $m : A \to A \oslash A$ satisfying some coherence conditions (associativity and identity). Note that morphisms of commutative comonoids in a sequoidal closed category do not admit a similar treatment (one would like to consider maps $f : (A, d_A) \to (B, d_B)$ such that $d_A \circ f = (f \oslash f) \circ d_B$, but this is only well-defined for strict f).

We next speak of cofree commutative comonoids. Given a symmetric monoidal category $\mathcal{C}$ one can construct the category of commutative comonoids $\mathsf{comonoids}(\mathcal{C})$, where an object is a commutative comonoid and a morphism is a comonoid morphism. There is an evident forgetful functor $U : \mathsf{comonoids}(\mathcal{C}) \to \mathcal{C}$ and one may ask if this functor has a right adjoint F: if it does, then $F(A)$ is the *cofree* commutative comonoid over A. This states that for each A there is a commutative comonoid $m_{F(A)} : F(A) \to F(A) \otimes F(A)$ and map $\eta_A : F(A) \to A$, such that for any commutative comonoid $m_B : B \to B \otimes B$ and map $f : B \to A$ there is a unique morphism $f^\dagger : B \to F(A)$ such

that $m_{F(A)} \circ f^\dagger = (f^\dagger \otimes f^\dagger) \circ m_B$ and $f = \eta_A \circ f^\dagger$.



In the case that $F(A) = (!A, \mathsf{d})$ then dereliction is η_A and promotion is $f \mapsto f^\dagger$. The sequoidal exponential is the cofree commutative comonoid in $\mathcal{G}$ and $\mathcal{W}$ [51].

Boolean Cell Strategy

Since our strategies are history-sensitive, we can construct strategies on $!N$ that are not merely the promotion of a strategy on N. For example, let $\mathbf{B}$ be the game of Booleans $\bot \lhd (\top \oplus \top)$ (we will call the opening move q and its two responses `tt` and `ff` — a dialogue consists of Opponent asking for a Boolean, and then Player giving him one). Then there is a strategy on $!\mathbf{B}$ which alternates between `tt` and `ff` responses:

$!\mathbf{B}$	
q	O
`tt`	P
q	O
`ff`	P
q	O
`tt`	P
⋮	

More elaborately, let $\mathbf{Bi} = (\bot \& \bot) \lhd \top$ (here Opponent gives an input `tt` or `ff` and Player responds with a). Then $!\mathbf{B} \otimes !\mathbf{Bi} \cong !(\mathbf{B} \& \mathbf{Bi})$ represents the type of a Boolean storage cell: it has a 'read' method (where Opponent asks for a Boolean and Player gives him one) and a 'write' method (where Opponent specifies a value to be written, and Player gives a confirmation). The exponential ensures that these methods can be 'called' arbitrarily many times. Thus we have a strategy cell on this type representing a reusable Boolean cell. This strategy is used to give the semantics of the new-variable constructor of Idealized Algol in the fully abstract game semantics given in [8]. An example play is as follows:

!(**B**	&	**Bi**)	
		`tt`	O
		a	P
q			O
`tt`			P
q			O
`tt`			P
		`ff`	O
		a	P
q			O
`ff`			P
⋮			

Here the behaviour of the strategy in each copy depends on what has previously happened in other copies (in the diagram above, all copies are compressed into a single column, the moves in odd positions each open a new copy).

The above discussion is specific to our choice of exponential, and demonstrates how it is well-suited for modelling imperative behaviour. The key isomorphism is $!N \cong N \oslash !N$.

3.3.2 Final Coalgebra

Using our sequoidal operator $\oslash$, we can describe a canonical property that our specific exponential $!N$ satisfies. As well as being the cofree commutative comonoid [51,62] it is the final coalgebra of the functor $X \mapsto N \oslash X$.

Final Coalgebras

Recall that a *coalgebra* for a functor $F : \mathcal{C} \to \mathcal{C}$ is an object A and a map $A \to F(A)$. A *final coalgebra* is a terminal object in the category of coalgebras, that is a coalgebra $\alpha : Z \to F(Z)$ such that for any $f : A \to F(A)$ there is a unique $(\!| f |\!) : A \to Z$ such that $\alpha \circ (\!| f |\!) = F((\!| f |\!)) \circ f$.

$$\begin{array}{ccc} A & \xrightarrow{f} & F(A) \\ {\scriptstyle (\!| f |\!)}\downarrow & & \downarrow{\scriptstyle F((\!| f |\!))} \\ Z & \xrightarrow[\alpha]{} & F(Z) \end{array}$$

We call $(\!|\, f \,|\!)$ the *anamorphism* of f. There are some standard properties of anamorphisms [74]: as well as $\sigma = (\!|\, \phi \,|\!)$ if and only if $\alpha \circ \sigma = F(\sigma) \circ \phi$ the following hold:

- Cancellation — $\alpha \circ (\!|\, \phi \,|\!) = F((\!|\, \phi \,|\!)) \circ \phi$
- Reflection — $\mathsf{id} = (\!|\, \alpha \,|\!)$
- Fusion — $\phi \circ \sigma = F(\sigma) \circ \psi \Rightarrow (\!|\, \phi \,|\!) \circ \sigma = (\!|\, \psi \,|\!)$
- Isomorphism — $\alpha^{-1} = (\!|\, F(\alpha) \,|\!)$.

Exponential as a Final Coalgebra

We define the map $\alpha : !N \to N \oslash !N$ by the copycat strategy $\mathsf{wk} \circ (\mathsf{der} \otimes \mathsf{id}) \circ \mathsf{d}$. This relabels $\mathsf{in}_1(a)$ on the right to $(a, 1)$ on the left and $\mathsf{in}_2(a, n)$ on the right to $(a, n+1)$ on the left.

Proposition 3.3.1 *$(!N, \alpha)$ is the final coalgebra of the functor $N \oslash \_$ in the category $\mathcal{G}$.*

Proof Let $\sigma : M \to N \oslash M$. Define $(\!|\, \sigma \,|\!)_n : M \to (N \oslash \_)^n(M)$ by $(\!|\, \sigma \,|\!)_0 = \mathsf{id}$ and $(\!|\, \sigma \,|\!)_{n+1} = (\mathsf{id} \oslash \_)^n(\sigma) \circ (\!|\, \sigma \,|\!)_n$.

$$M \xrightarrow{(\!|\, \sigma \,|\!)_n} (N \oslash \_)^n(M) \xrightarrow{(\mathsf{id} \oslash \_)^n(\sigma)} (N \oslash \_)^n(N \oslash M) = (N \oslash \_)^{n+1}(M)$$

The strategy $(\!|\, \sigma \,|\!)_n$ is a partial approximant to $(\!|\, \sigma \,|\!) : M \to !N$. We can show by induction on n that $(\!|\, \sigma \,|\!)_{n+1} = (\mathsf{id} \oslash (\!|\, \sigma \,|\!)_n) \circ \sigma$. For $n = 0$ we have $(\!|\, \sigma \,|\!)_1 = (\mathsf{id} \oslash \_)^0(\sigma) \circ \mathsf{id} = \sigma = (\mathsf{id} \oslash \mathsf{id}) \circ \sigma = (\mathsf{id} \oslash (\!|\, \sigma \,|\!)_0) \circ \sigma$ as required. For $n = m + 1$ we have $(\!|\, \sigma \,|\!)_{m+2} = (\mathsf{id} \oslash \_)^{m+1}(\sigma) \circ (\!|\, \sigma \,|\!)_{m+1} = (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^m(\sigma)) \circ (\mathsf{id} \oslash (\!|\, \sigma \,|\!)_m) \circ \sigma = (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^m(\sigma) \circ (\!|\, \sigma \,|\!)_m) \circ \sigma = (\mathsf{id} \oslash (\!|\, \sigma \,|\!)_{m+1}) \circ \sigma$ as required.

Similarly, we can define $\alpha_k : !N \cong (N \oslash \_)^k(!N) : \alpha_k^{-1}$ by performing the above construction on α. Consider the sequence of maps $M \to !N$ defined by $s_k = \alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ (\!|\, \sigma \,|\!)_k$ for $k \in \omega$. We show that $s_{k+1} \sqsupseteq s_k$ and so (s_k) is a chain. For $k = 0$, the RHS is ϵ and so we are done. Otherwise, $s_{k+2} = \alpha_{k+2}^{-1} \circ (\mathsf{id} \oslash \_)^{k+2}(\epsilon) \circ (\!|\, \sigma \,|\!)_{k+2} = \alpha^{-1} \circ (\mathsf{id} \oslash \alpha_{k+1}^{-1}) \circ (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^{k+1}(\epsilon)) \circ (\mathsf{id} \oslash (\!|\, \sigma \,|\!)_{k+1}) \circ \sigma = \alpha^{-1} \circ (\mathsf{id} \oslash \alpha_{k+1}^{-1} \circ (\mathsf{id} \oslash \_)^{k+1}(\epsilon) \circ (\!|\, \sigma \,|\!)_{k+1}) \circ \sigma \sqsubseteq \alpha^{-1} \circ (\mathsf{id} \oslash \alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ (\!|\, \sigma \,|\!)_k) \circ \sigma = \alpha_{k+1}^{-1} \circ (\mathsf{id} \oslash \_)^{k+1}(\epsilon) \circ (\!|\, \sigma \,|\!)_{k+1}$.

Set $(\!|\, \sigma \,|\!) = \bigsqcup \alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ (\!|\, \sigma \,|\!)_k$, where ϵ is the empty strategy. It is well-known that $\mathcal{G}$ is cpo-enriched with bottom element ϵ [51].

We wish to show that $(\!|\, \sigma \,|\!)$ is the unique strategy such that $\alpha \circ (\!|\, \sigma \,|\!) = (\mathsf{id} \oslash (\!|\, \sigma \,|\!)) \circ \sigma$. To show that the equation holds, note that $\alpha \circ (\!|\, \sigma \,|\!) = \alpha \circ \bigsqcup \alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ (\!|\, \sigma \,|\!)_k =$

$\alpha \circ \bigsqcup \alpha_{k+1}^{-1} \circ (\mathsf{id} \oslash \_)^{k+1}(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_{k+1} = \bigsqcup \alpha \circ \alpha_{k+1}^{-1} \circ (\mathsf{id} \oslash \_)^{k+1}(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_{k+1} = \bigsqcup (\mathsf{id} \oslash \alpha_k^{-1}) \circ (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^k(\epsilon)) \circ (\mathsf{id} \oslash \mathbb{(} \sigma \mathbb{)}_k) \circ \sigma = (\mathsf{id} \oslash \bigsqcup (\alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_k) \circ \sigma = (\mathsf{id} \oslash \mathbb{(} \sigma \mathbb{)}) \circ \sigma$.

For uniqueness, suppose that $\gamma : M \to !N$ is such that $\alpha \circ \gamma = (\mathsf{id} \oslash \gamma) \circ \sigma$. We wish to show that $\gamma = \mathbb{(} \sigma \mathbb{)} = \bigsqcup \alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_k$. To see that $\gamma \sqsupseteq \mathbb{(} \sigma \mathbb{)}$, it suffices to show that γ is an upper bound of the chain, i.e. $\gamma \sqsupseteq \alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_k$ for each k. We proceed by induction on k. For $k = 0$ we are done, as the RHS is ϵ. Then $\alpha_{k+1}^{-1} \circ (\mathsf{id} \oslash \_)^{k+1}(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_{k+1} = \alpha^{-1} \circ (\mathsf{id} \oslash \alpha_k^{-1}) \circ (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^k(\epsilon)) \circ (\mathsf{id} \oslash \mathbb{(} \sigma \mathbb{)}_k) \circ \sigma \sqsubseteq \alpha^{-1} \circ (\mathsf{id} \oslash \gamma) \circ \sigma = \gamma$. Hence γ is an upper bound of this chain, so $\gamma \sqsupseteq \mathbb{(} \sigma \mathbb{)}$.

To see that $\gamma \sqsubseteq \mathbb{(} \sigma \mathbb{)}$, we show that each play in γ is also in $\mathbb{(} \sigma \mathbb{)}$. Consider a play $s \in \gamma : M \to !N$. Since s is finite, it must visit only a finite number of copies of N — say, k copies. Then s is also a play in $\alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ \alpha_k \circ \gamma$. If we show that $\alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ \alpha_k \circ \gamma \sqsubseteq \alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_k$ then we will be done, as we would have shown that s is a play in one of the finite approximates of $\mathbb{(} \sigma \mathbb{)}$, and so $s \in \mathbb{(} \sigma \mathbb{)}$.

It is thus sufficient to show that $(\mathsf{id} \oslash \_)^k(\epsilon) \circ \alpha_k \circ \gamma \sqsubseteq (\mathsf{id} \oslash \_)^k(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_k$. We show this by induction on k. If $k = 0$, then the right hand side is ϵ and so we are done. If $k = j + 1$ then $(\mathsf{id} \oslash \_)^{j+1}(\epsilon) \circ \alpha_{j+1} \circ \gamma = (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^j(\epsilon)) \circ (\mathsf{id} \oslash \alpha_j) \circ \alpha \circ \gamma = (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^j(\epsilon)) \circ (\mathsf{id} \oslash \alpha_j) \circ (\mathsf{id} \oslash \gamma) \circ \sigma = (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^j(\epsilon) \circ \alpha_j \circ \gamma) \circ \sigma \sqsubseteq (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^j(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_j) \circ \sigma = (\mathsf{id} \oslash \_)^{j+1}(\epsilon) \circ (\mathsf{id} \oslash \mathbb{(} \sigma \mathbb{)}_j) \circ \sigma = (\mathsf{id} \oslash \_)^{j+1}(\epsilon) \circ \mathbb{(} \sigma \mathbb{)}_{j+1}$ as required. ∎

Note that the above construction follows from the fact that $!N$ is the minimal invariant of $N \oslash \_$ in $\mathcal{G}$ [51], from which it also follows that $!N$ is also the initial algebra of the same in $\mathcal{G}$. This stronger property does not hold in $\mathcal{W}$, although the final coalgebra property does. This depends on the choice of winning condition for $!$, and alternative winning conditions on the same game have been used elsewhere to move between initial algebras and final coalgebras of a given functor [20].

Proposition 3.3.2 $(!N, \alpha)$ *is the final coalgebra of* $N \oslash \_$ *in the category* $\mathcal{W}$.

Proof The construction given above can take place in $\mathcal{W}$. Given winning $\sigma : M \to N \oslash M$ we only need to check that $\mathbb{(} \sigma \mathbb{)}$ is winning.

To see that $\mathbb{(} \sigma \mathbb{)}$ is total, let $s \in \mathbb{(} \sigma \mathbb{)}$ and $so \in P_{!N}$. Then so visits only finite k many copies of N, and so up to retagging it is a play in $M \to (N \oslash \_)^k(M)$, and s a play in $\mathbb{(} \sigma \mathbb{)}_k$. By totality of $\mathbb{(} \sigma \mathbb{)}_k$, there is a move p with $sop \in \mathbb{(} \sigma \mathbb{)}_k$. Then, up to retagging, sop is also a play in $\mathbb{(} \sigma \mathbb{)}$.

We next need to check that each infinite play with all even prefixes in $\mathbb{(} \sigma \mathbb{)}$ is winning. Let s be such an infinite play, with $s|_M$ winning. We must show that $s|_{!N}$ is winning, i.e. $s|_{(N,i)}$ is winning for each i. The infinite play s corresponds to an infinite interaction sequence:

$$M \xrightarrow{\sigma} N \oslash M \xrightarrow{\mathsf{id} \oslash \sigma} N \oslash (N \oslash M) \xrightarrow{\mathsf{id} \oslash (\mathsf{id} \oslash \sigma)} \ldots$$

$$\vdots$$

Then $s|_{(N,i)}$ can also be found in the ith column of the above interaction sequence. By hiding all columns other than the first and the ith, we see a play in $M \to (N \oslash \_)^i(M)$ in $(\!|\, \sigma \,|\!)_i$. The first column is $s|_M$ (which is winning), and the ith component of the second is $s|_{(N,i)}$. Since since $(\!|\, \sigma \,|\!)_i$ is a winning strategy, this play is winning, by the winning condition for $\oslash$. ∎

We will later need a further coalgebraic property of the operation ! in $\mathcal{G}$, following the observations above.

Proposition 3.3.3 *Suppose $\phi : M \to N \oslash M$, $\sigma : M \to !N$ and $\psi : !N \to N \oslash !N$. 1. If $\alpha \circ \sigma \sqsupseteq (\mathsf{id} \oslash \sigma) \circ \phi$ then $\sigma \sqsupseteq (\!|\, \phi \,|\!)$. 2. If $(\mathsf{id} \oslash \sigma) \circ \phi \sqsubseteq \psi \circ \sigma$ then $(\!|\, \phi \,|\!) \sqsubseteq (\!|\, \psi \,|\!) \circ \sigma$.*

Proof 1. Suppose $\alpha \circ \sigma \sqsupseteq (\mathsf{id} \oslash \sigma) \circ \phi$. Then $\sigma \sqsupseteq \alpha^{-1} \circ (\mathsf{id} \oslash \sigma) \circ \phi$. To show that $\sigma \sqsupseteq (\!|\, \phi \,|\!)$, it suffices to show that for all k, $\sigma \sqsupseteq \alpha_k^{-1} \circ (\mathsf{id} \oslash \_)^k(\epsilon) \circ (\!|\, \phi \,|\!)_k$. For $k = 0$ we are done as the RHS is ϵ. For $k = j + 1$, note that $\alpha_{j+1}^{-1} \circ (\mathsf{id} \oslash \_)^{j+1}(\epsilon) \circ (\!|\, \phi \,|\!)_{j+1} = \alpha^{-1} \circ (\mathsf{id} \oslash \alpha_j^{-1}) \circ (\mathsf{id} \oslash (\mathsf{id} \oslash \_)^j(\epsilon)) \circ (\mathsf{id} \oslash (\!|\, \phi \,|\!)_j) \circ \phi \sqsubseteq \alpha^{-1} \circ (\mathsf{id} \oslash \sigma) \circ \phi \sqsubseteq \sigma$.

2. To show that $(\!|\, \phi \,|\!) \sqsubseteq (\!|\, \psi \,|\!) \circ \sigma$ it is sufficient by 1 to show that $(\mathsf{id} \oslash (\!|\, \psi \,|\!) \circ \sigma) \circ \phi \sqsubseteq \alpha \circ (\!|\, \psi \,|\!) \circ \sigma$. Since $\alpha \circ (\!|\, \psi \,|\!) = (\mathsf{id} \oslash (\!|\, \psi \,|\!)) \circ \psi$ we need to show that $(\mathsf{id} \oslash (\!|\, \psi \,|\!) \circ \sigma) \circ \phi \sqsubseteq (\mathsf{id} \oslash (\!|\, \psi \,|\!)) \circ \psi \circ \sigma$. But this is clear as $(\mathsf{id} \oslash (\!|\, \psi \,|\!) \circ \sigma) \circ \phi = (\mathsf{id} \oslash (\!|\, \psi \,|\!)) \circ (\mathsf{id} \oslash \sigma) \circ \phi \sqsubseteq (\mathsf{id} \oslash (\!|\, \psi \,|\!)) \circ \psi \circ \sigma$ as required. ∎

Boolean Cell and Stack Example

We can express our Boolean cell strategy as the anamorphism of a finite strategy. Define a parametrised cell $\mathsf{cell}' : \mathbf{B} \to !(\mathbf{B} \& \mathbf{Bi})$ as $(\!|\, f \,|\!)$ where $f : \mathbf{B} \to (\mathbf{B} \& \mathbf{Bi}) \oslash \mathbf{B}$. The strategy f is defined as follows, with b ranging over $\{\mathtt{tt}, \mathtt{ff}\}$:

$\mathbf{B} \multimap (\mathbf{B} \& \mathbf{Bi}) \oslash \mathbf{B}$			$\mathbf{B} \multimap (\mathbf{B} \& \mathbf{Bi}) \oslash \mathbf{B}$		
	q	O		b	
q		P		a	
b		O			q
	b	P			b
		q O			
		b P			

Then $\mathsf{cell} = \mathsf{cell}' \circ \mathbf{tf}$ where $\mathbf{tf}$ is the strategy on $\mathbf{B}$ specifying the starting value of the cell. We can see the parameter to cell' as being the explicitly propagated 'state' between the interactions in the subsequent copies of $\mathbf{B}\&\mathbf{Bi}$.

We can perform a similar treatment to construct a Boolean *stack*, with unbounded memory. We can construct this using the anamorphism of a strategy that is a finite sequence of moves followed by copycat. The type of the stack strategy is also $!(\mathbf{B}\&\mathbf{Bi})$ — the $\mathbf{B}$ component represents a 'pop' method and the $\mathbf{Bi}$ component a 'push' method. We can consider a strategy $\mathsf{stack}' : !\mathbf{B} \to !(\mathbf{B}\&\mathbf{Bi})$ which represents a stack parametrised by a 'starting stack' (the argument represents the behaviour that would be observed by running 'pop' an arbitrary number of times). We set $\mathsf{stack}' = (\!|\, g \,|\!)$ where $g : !\mathbf{B} \to (\mathbf{B}\&\mathbf{Bi}) \oslash !\mathbf{B}$ is as follows:

$!\mathbf{B}$	$\multimap$	$(\mathbf{B}$	$\&$	$\mathbf{Bi})$	$\oslash$	$!\mathbf{B}$
		q				O
q						P
b						O
		b				P
					q	O
q						P
b'						O
					b'	P
					$\vdots$	

$!\mathbf{B}$	$\multimap$	$(\mathbf{B}$	$\&$	$\mathbf{Bi})$	$\oslash$	$!\mathbf{B}$
				b		
				a		
						q
						b
						q
q						
b'						
						b'
						$\vdots$

After the first four moves the strategy enters copycat. The initial stack $\mathsf{s} : !\mathbf{B}$ can also be defined by an anamorphism. For example, a stack containing `ff` elements can be defined as $(\!|\, g \,|\!)$ where $g : I \to \mathbf{B} \oslash I$ contains the unique maximal play $q\mathtt{ff}$.

Deriving the Exponential Structure

Just as we can derive dereliction from the fact that $!N$ is the final coalgebra of $X \mapsto N \oslash X$, we might hope to derive contraction, promotion and multiplication as well. This does not seem to be possible, but starting from any one of the three the other two are derivable:

1. Contraction $\mathsf{d} : !N \to !N \otimes !N$ (or equivalently $\mathsf{con} : !N \to !N \oslash !N$)
2. Comonad multiplication $\mathsf{mult} : !N \to !!N$
3. Promotion $(-)^\dagger$ constructing a map $!N \to !M$ from a map $!N \to M$

For (1) $\Rightarrow$ (2) we can set $\mathsf{mult} = \mathbb{(}\ \mathsf{con}\ \mathbb{)}$. For (2) $\Rightarrow$ (1) we have $\mathsf{con} = (\mathsf{id} \oslash \mathsf{der}) \circ \alpha \circ \mathsf{mult}$. For (1) $\Rightarrow$ (3) we can set $\sigma^\dagger = \mathbb{(}\ \mathsf{wk} \circ (\sigma \otimes \mathsf{id}) \circ \mathsf{d}\ \mathbb{)}$. For (3) $\Rightarrow$ (2) we have $\mathsf{mult} = \mathsf{id}^\dagger$.

We can make use of this in our categorical axiomatics. If we assume that $!N$ is the carrier of a commutative comonoid with $\mathsf{d} : !N \to !N \otimes !N$, we can use the final coalgebra property to define the cofree commutative comonoid structure of $!N$. We can define η_A and $f^\dagger$ using the final coalgebraic and comonoid structure. Let $\eta_A = \mathsf{der} = \mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{t}) \circ \alpha$. If $m_B : B \to B \otimes B$ is a commutative comonoid and $f : B \to A$, then we set $f^\dagger = \mathbb{(}\ \mathsf{wk} \circ (f \otimes \mathsf{id}) \circ m_B\ \mathbb{)}$. The condition for $!$ to be the cofree commutative comonoid is that $f^\dagger : B \to !A$ is unique comonoid morphism such that $f = \eta_A \circ f^\dagger$. Thus, if this property is satisfied, our final coalgebra structure can be used to derive a cofree commutative comonoid structure and be an instance of a known categorical characterisation of the linear logic exponential.

Definition A sequoidal closed category $\mathcal{C}$ has a *coalgebraic exponential comonoid* if:

- The endofunctor $N \oslash \_$ on $\mathcal{C}$ has a final coalgebra $(!N, \alpha_N)$ where $\alpha_N \in \mathcal{C}_s$. Let $\mathsf{der} : !A \to A$ be $\mathsf{unit}_\oslash \circ (\mathsf{id} \oslash \mathsf{t}) \circ \alpha$
- For each N, $!N$ is the carrier of a commutative comonoid whose multiplication d is strict and $\mathsf{wk} \circ (\mathsf{der} \otimes \mathsf{id}) \circ \mathsf{d} = \alpha$

$$\begin{array}{ccc} !A & \xrightarrow{\alpha} & A \oslash !A \\ {\scriptstyle \mathsf{d}}\downarrow & & \uparrow{\scriptstyle \mathsf{wk}} \\ !A \otimes !A & \xrightarrow{\mathsf{der} \otimes \mathsf{id}} & A \otimes !A \end{array}$$

- For each commutative comonoid $m : B \to B \otimes B$ and $f : B \to A$, $f^\dagger = \mathbb{(}\ \mathsf{wk} \circ (f \otimes \mathsf{id}) \circ m\ \mathbb{)} : B \to !A$ is the unique comonoid morphism such that $f = \mathsf{der} \circ f^\dagger$.

A coalgebraic exponential comonoid is a cofree commutative comonoid, and so can be used to model the exponential modality.

We pause to note that using the above construction $!f = (f \circ \mathsf{der})^\dagger = \mathbb{(}\ \mathsf{wk} \circ (f \circ \mathsf{der} \otimes \mathsf{id}) \circ \mathsf{d}\ \mathbb{)}$. In the case that f is strict, this is $\mathbb{(}\ (f \oslash \mathsf{id}) \circ \mathsf{wk} \circ (\mathsf{der} \otimes \mathsf{id}) \circ \mathsf{d}\ \mathbb{)} = \mathbb{(}\ (f \oslash \mathsf{id}) \circ \alpha\ \mathbb{)}$. Also, the family of maps $\mathsf{d} : !N \to !N \otimes !N$ is natural in N: this is equivalent to $!f$ being a comonoid morphism, which holds using the third condition and the above expansion of $!f$.

Proposition 3.3.4 *The sequoidal closed categories $\mathcal{W}$ and $\mathcal{G}$ are both equipped with a coalgebraic exponential comonoid.*

Proof Follows from Propositions 3.3.1, 3.3.2 and the fact ! is the cofree commutative comonoid in $\mathcal{G}$ and $\mathcal{W}$ [51]. The coherence condition $\mathsf{wk} \circ (\mathsf{der} \otimes \mathsf{id}) \circ \mathsf{d} = \alpha$ can be easily checked. ∎

3.4 The Logic WS!

We next add the exponential operators to WS, with proof rules explicitly based on the fact that $!N$ is a final coalgebra. We call this extended system WS!.

3.4.1 Proof System

Formulas of WS!

The formulas of WS! are those of WS extended with exponentials:

$$\begin{array}{rcccccccccccc} P & := & \mathbf{0} & | & \bot & | & P \parr Q & | & P \oplus Q & | & P \lhd Q & | & P \oslash N & | & ?P \\ N & := & \mathbf{1} & | & \top & | & N \otimes M & | & M \& N & | & N \oslash M & | & N \lhd P & | & !N \end{array}$$

The new exponential operators will be interpreted by the exponential modalities in Section 3.3.1.

Proof Rules

Sequents of WS! are again lists of formulas, and the proof rules of WS! are those of WS extended by the additions in Figure 3-1.

Figure 3-1: Proof rules for WS! — extends Figure 2-2

Core rules:

$$\mathsf{P}_! \frac{\vdash N, !N, \Gamma}{\vdash !N, \Gamma} \qquad \mathsf{P}_? \frac{\vdash P, ?P, \Gamma}{\vdash ?P, \Gamma}$$

Other rules:

$$\mathsf{P}^!_{\mathsf{der}} \frac{\vdash \Gamma, !M, \Delta}{\vdash \Gamma, M, \Delta} \qquad \mathsf{P}^!_{\mathsf{con}} \frac{\vdash \Gamma, !M, \Delta}{\vdash \Gamma, !M, !M, \Delta} \qquad \mathsf{P}_{\mathsf{ana}} \frac{\vdash M, P^\perp, P}{\vdash !M, P}$$

$$\mathsf{P}^?_{\mathsf{der}} \frac{\vdash \Gamma, P, \Delta}{\vdash \Gamma, ?P, \Delta} \qquad \mathsf{P}^?_{\mathsf{con}} \frac{\vdash \Gamma, ?P, ?P, \Delta}{\vdash \Gamma, ?P, \Delta}$$

We can interpret the new rules as operations on strategies as follows:

- The $\mathsf{P}_!$ rule is interpreted by noting that a strategy on the premise corresponds to a strategy on the conclusion, via the isomorphism $!N \cong N \oslash !N$.

- The $\mathsf{P}_?$ rule is interpreted by noting that a strategy on the premise corresponds to a strategy on the conclusion, via the isomorphism $?P \cong P \lhd ?P$.
- The $\mathsf{P}^!_{\mathsf{der}}$ and $\mathsf{P}^?_{\mathsf{der}}$ rules are interpreted by composition with the dereliction map $\mathsf{der} : !\,M \multimap M$ which uses only the first copy on the left.
- The $\mathsf{P}^!_{\mathsf{con}}$ and $\mathsf{P}^?_{\mathsf{con}}$ rules are interpreted by composition with the contraction map $\mathsf{con} : !\,M \multimap !\,M \otimes !M$ which opens a new copy on the left for each copy in either component on the right.
- The $\mathsf{P}_{\mathsf{ana}}$ rule is interpreted by using the final coalgebra property of $!\,N$, as described above.

3.4.2 Embedding ILL in WS!

We have already seen how we can use anamorphisms and contraction to define promotion. This is reflected in the logic, as we can define promotion as a derived rule in WS!. We can extend Proposition 2.2.1 to full Intuitionistic Linear Logic (over the units $\bot$ and $\mathbf{1}$). The proof rules of ILL are those of IMALL in Figure 2-3 together with the exponential rules given in Figure 3-2 [72] ($!\Gamma$ ranges over contexts where every element is of the form $!M$).

Figure 3-2: Proof rules for ILL — extends Figure 2-3

$$\frac{\Gamma, N \vdash M}{\Gamma, !N \vdash M} \qquad \frac{\Gamma, !N, !N \vdash M}{\Gamma, !N \vdash M} \qquad \frac{\Gamma \vdash M}{\Gamma, !N \vdash M} \qquad \frac{!\Gamma \vdash N}{!\Gamma \vdash !N}$$

Proposition 3.4.1 *For each proof p of $M_1, \ldots, M_n \vdash N$ in ILL there is a proof $\kappa(p)$ in WS! of $\vdash N, M_1^\bot, \ldots, M_n^\bot$.*

Proof We have already described the translation of the exponential-free fragment in Proposition 2.2.1.

The first three exponential rules correspond to $\mathsf{P}^?_{\mathsf{der}}$, $\mathsf{P}^?_{\mathsf{con}}$ and $\mathsf{P}^+_{\mathsf{wk}}$ respectively. We next give the translation of the right ! rule (promotion). This makes use of the explicit anamorphisms found in WS!. We first assume Γ consists of a single formula L.

$$\mathsf{P}_{\mathsf{mul}}\,\frac{\vdash N, ?L^\bot \qquad \mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash !L, ?L^\bot}}{\mathsf{P}^?_{\mathsf{con}}\,\dfrac{\vdash N, !L, ?L^\bot, ?L^\bot}{\mathsf{P}_{\mathsf{ana}}\,\dfrac{\vdash N, !L, ?L^\bot}{\vdash !N, ?L^\bot}}}$$

We will later refer to this derived rule as $\mathsf{P}_{\mathsf{prom}}$. If Γ contains more than one formula, we use the equivalence of $!M\otimes!N$ and $!(M\&N)$ in $\mathsf{WS!}$.

The first direction $p_1 \vdash !M\otimes!N \multimap !(M\&N)$ is defined as follows:

$$\mathsf{P}_{\mathsf{ana}}\ \dfrac{\mathsf{P}_{\&}\ \dfrac{\mathsf{P}^{\mathsf{T}}_{\otimes}\ \dfrac{\mathsf{P}^{\mathsf{T}}_{\parr}\ \dfrac{\mathsf{P}^{!}_{\mathsf{der}}\ \dfrac{\mathsf{P}^{!}_{\mathsf{con}}\ \dfrac{\mathsf{P}_{\mathsf{mul}}\ \dfrac{\mathsf{P}_{\mathsf{id}}\ \dfrac{}{\vdash !M, ?M^\perp} \qquad \mathsf{P}_{\mathsf{id}}\ \dfrac{}{\vdash !N, ?N^\perp}}{\vdash !M, !N, ?M^\perp, ?N^\perp}}{\vdash !M, !M, !N, ?M^\perp, ?N^\perp}}{\vdash M, !M, !N, ?M^\perp, ?N^\perp}}{\vdash M, !M, !N, ?M^\perp \parr ?N^\perp}}{\vdash M, !M\otimes!N, ?M^\perp \parr ?N^\perp} \qquad \mathsf{P}^{\mathsf{T}}_{\otimes}\ \dfrac{\mathsf{P}^{\mathsf{T}}_{\parr}\ \dfrac{\mathsf{P}^{!}_{\mathsf{der}}\ \dfrac{\mathsf{P}_{\mathsf{sym}}\ \dfrac{\mathsf{P}^{!}_{\mathsf{con}}\ \dfrac{\mathsf{P}_{\mathsf{mul}}\ \dfrac{\mathsf{P}_{\mathsf{id}}\ \dfrac{}{\vdash !M, ?M^\perp} \qquad \mathsf{P}_{\mathsf{id}}\ \dfrac{}{\vdash !N, ?N^\perp}}{\vdash !M, !N, ?M^\perp, ?N^\perp}}{\vdash !M, !N, !N, ?M^\perp, ?N^\perp}}{\vdash !N, !M, !N, ?M^\perp, ?N^\perp}}{\vdash N, !M, !N, ?M^\perp, ?N^\perp}}{\vdash N, !M, !N, ?M^\perp \parr ?N^\perp}}{\vdash N, !M\otimes!N, ?M^\perp \parr ?N^\perp}}{\vdash M\&N, !M\otimes!N, ?M^\perp \parr ?N^\perp}}{\vdash !(M\&N), ?M^\perp \parr ?N^\perp}$$

The second direction $p_2 \vdash !(M\&N) \multimap !M\otimes!N$ is given as follows:

$$\mathsf{P}^{?}_{\mathsf{con}}\ \dfrac{\mathsf{P}_{\mathsf{mul}\otimes}\ \dfrac{\mathsf{P}_{\mathsf{prom}}\ \dfrac{\mathsf{P}^{?}_{\mathsf{der}}\ \dfrac{\mathsf{P}^{\mathsf{T}}_{\oplus 1}\ \dfrac{\mathsf{P}_{\mathsf{id}}\ \dfrac{}{\vdash M, M^\perp}}{\vdash M, M^\perp \oplus N^\perp}}{\vdash M, ?(M^\perp \oplus N^\perp)}}{\vdash !M, ?(M^\perp \oplus N^\perp)} \qquad \mathsf{P}_{\mathsf{prom}}\ \dfrac{\mathsf{P}^{?}_{\mathsf{der}}\ \dfrac{\mathsf{P}^{\mathsf{T}}_{\oplus 2}\ \dfrac{\mathsf{P}_{\mathsf{id}}\ \dfrac{}{\vdash N, N^\perp}}{\vdash N, M^\perp \oplus N^\perp}}{\vdash N, ?(M^\perp \oplus N^\perp)}}{\vdash !N, ?(M^\perp \oplus N^\perp)}}{\vdash !M\otimes!N, ?(M^\perp \oplus N^\perp), ?(M^\perp \oplus N^\perp)}}{\vdash !M\otimes!N, ?(M^\perp \oplus N^\perp)}$$

The interpretations of these maps are the standard copycat morphisms in $\mathcal{W}$.

We can then generalise $\mathsf{P}_{\mathsf{prom}}$ to

$$\mathsf{P}^{\mathsf{T}}_{\parr}\ \dfrac{\mathsf{P}_{\mathsf{cut}}\ \dfrac{\dfrac{\vdots}{\vdash !M, ?(P_1 \oplus P_2), \ldots, ?P_{n-1}, ?P_n}\ \dfrac{\mathsf{P}_{\mathsf{prom}}\ \dfrac{\mathsf{P}_{\mathsf{cut}}\ \dfrac{\mathsf{P}^{\mathsf{T}}_{\parr}\ \dfrac{\dfrac{\dfrac{\vdash M, ?P_1, ?P_2, \ldots, ?P_{n-1}, ?P_n}{\vdots}}{\vdash M, ?(P_1 \oplus P_2 \oplus \ldots \oplus P_{n-1}), ?P_n}}{\vdash M, ?(P_1 \oplus P_2 \oplus \ldots \oplus P_{n-1}) \parr ?P_n} \qquad p_2}{\vdash M, ?(P_1 \oplus P_2 \oplus \ldots \oplus P_{n-1} \oplus P_n)}}{\vdash !M, ?(P_1 \oplus P_2 \oplus \ldots \oplus P_{n-1} \oplus P_n)}}{\vdots} \qquad p_1}{\vdash !M, ?P_1 \parr ?P_2, \ldots, ?P_{n-1}, ?P_n}}{\vdash !M, ?P_1, ?P_2, \ldots, ?P_{n-1}, ?P_n}$$

and interpret the right ! rule of ILL. ∎

3.4.3 Boolean Cell and Stack

We will next show how some infinitary imperative objects can be expressed as proofs in WS!.

A Boolean Cell

We next give a proof that denotes our Boolean cell strategy **cell** :!(**B**&**Bi**) from Section 3.3.2. We first define **cell**′ : **B** ⊸!(**B**&**Bi**) giving the behaviour of the cell parametrised by a given starting value. In particular, we take the anamorphism of a map **B** ⊸ (**B**&**Bi**) ⅋ **B**. This proof is given in Figure 3-3. Its semantics is the history-sensitive Boolean cell strategy given in [8]. The proof $p_{\texttt{read}}$ corresponds to the map **B** ⊸ **B** ⅋ **B** which reads its argument and propagates it to the next call, and $p_{\texttt{write}}$ corresponds to the map **B** ⊸ **Bi** ⅋ **B** which ignores its argument and propagates the written value to the next call.

We can use **cell** (together with the ILL embedding) to give an embedding of recursion-free Idealized Algol with finitary datatypes into WS!. We will explore this theme in Chapter 5, also considering other imperative programming constructs that can be represented by $\mathsf{P}_{\mathsf{ana}}$.

A Boolean Stack

We can similarly give a proof denoting our Boolean stack in Section 3.3.2. We first define a proof !**B** ⊸!(**B**&**Bi**), representing a parametrised stack as above. This is given in Figure 3-4. An initial stack (proof of !**B**) can be defined easily using $\mathsf{P}_{\mathsf{prom}}$.

3.4.4 Embedding LLP in WS!

In Section 2.6 we embedded MALLP inside WS. Full Polarized Linear Logic (LLP) can be embedded in WS!. LLP extends MALLP with exponential connectives:

$$\begin{array}{rcccccccccc} P & := & \mathbf{1} & | & \mathbf{0} & | & P \otimes Q & | & P \oplus Q & | & \downarrow N & | & !N \\ N & := & \bot & | & \top & | & M \parr N & | & M \& N & | & \uparrow P & | & ?P \end{array}$$

Remark The presentation of LLP given in [53] omits the linear lifts ↑ and ↓ of MALLP. We will include them in our presentation of LLP and its embedding, for use in Chapter 5.

We say a negative LLP formula N is *reusable* (and write $\mathsf{reuse}(N)$) if every occurrence of ↑ occurs under a ?. If we exclude the linear lifts ↑ and ↓, all negative formulas are

Figure 3-3: Proof Denoting a Boolean Cell

$$\mathsf{P}_{\mathsf{ana}}\ \frac{\overset{p_{\mathsf{write}}}{\cdots}\qquad\overset{p_{\mathsf{read}}}{\cdots}}{\vdash ((\bot\&\bot)\lhd\top)\&(\bot\lhd(\top\oplus\top)),\bot\lhd(\top\oplus\top),\top\oplus(\bot\&\bot)}}{\vdash !(((\bot\&\bot)\lhd\top)\&(\bot\lhd(\top\oplus\top))),\top\oplus(\bot\&\bot)}$$

where p_{write} is

$$\dfrac{\dfrac{\mathsf{P}_{\oplus 1}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash\top}}{\vdash\top,(\top\oplus(\bot\&\bot))}}{\vdash\top\oplus\top,(\top\oplus(\bot\&\bot))}}{\vdash(\top\oplus\top)\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot,(\top\oplus\top)\parr(\top\oplus(\bot\&\bot))}}{\vdash(\bot\lhd(\top\oplus\top))\lhd(\top\oplus(\bot\&\bot))}}{\vdash\top,(\bot\lhd(\top\oplus\top))\lhd(\top\oplus(\bot\&\bot))}}{\vdash(\top\oplus(\bot\lhd(\top\oplus\top))),\top\oplus(\bot\&\bot)}}{\vdash(\top\oplus(\bot\lhd(\top\oplus\top)))\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot,(\top\oplus(\bot\lhd(\top\oplus\top)))\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot,\top,\bot\lhd(\top\oplus\top),\top\oplus(\bot\&\bot)}\qquad \mathsf{P}_{\oplus 2}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash\top}}{\vdash\top,(\top\oplus(\bot\&\bot))}}{\vdash\top\oplus\top,(\top\oplus(\bot\&\bot))}}{\vdash(\top\oplus\top)\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot,(\top\oplus\top)\parr(\top\oplus(\bot\&\bot))}}{\vdash(\bot\lhd(\top\oplus\top))\lhd(\top\oplus(\bot\&\bot))}}{\vdash\top,(\bot\lhd(\top\oplus\top))\lhd(\top\oplus(\bot\&\bot))}}{\vdash(\top\oplus(\bot\lhd(\top\oplus\top))),\top\oplus(\bot\&\bot)}}{\vdash(\top\oplus(\bot\lhd(\top\oplus\top)))\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot,(\top\oplus(\bot\lhd(\top\oplus\top)))\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot,\top,\bot\lhd(\top\oplus\top),\top\oplus(\bot\&\bot)}}{\vdash\bot\&\bot,\top,\bot\lhd(\top\oplus\top),\top\oplus(\bot\&\bot)}}{\vdash(\bot\&\bot)\lhd\top,\bot\lhd(\top\oplus\top),\top\oplus(\bot\&\bot)}$$

and p_{read} is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash\top}}{\vdash\top\oplus\top}}{\vdash\bot,(\top\oplus\top)}}{\vdash\top,\bot\lhd(\top\oplus\top)}}{\vdash\top\oplus\top,\bot\lhd(\top\oplus\top)}}{\vdash\bot,(\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top))}}{\vdash\bot\&\bot,(\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top))}}{\vdash\top,(\bot\&\bot)\lhd((\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top)))}}{\vdash\top\oplus(\bot\&\bot),(\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top))}}{\vdash((\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top)))\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot,((\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top)))\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot\lhd(\top\oplus\top),\bot\lhd(\top\oplus\top),\top\oplus(\bot\&\bot)}\qquad \mathsf{P}_{\oplus 2}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash\top}}{\vdash\top\oplus\top}}{\vdash\bot,(\top\oplus\top)}}{\vdash\top,\bot\lhd(\top\oplus\top)}}{\vdash\top\oplus\top,\bot\lhd(\top\oplus\top)}}{\vdash\bot,(\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top))}}{\vdash\bot\&\bot,(\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top))}}{\vdash\top,(\bot\&\bot)\lhd((\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top)))}}{\vdash\top\oplus(\bot\&\bot),(\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top))}}{\vdash((\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top)))\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot,((\top\oplus\top)\oslash(\bot\lhd(\top\oplus\top)))\parr(\top\oplus(\bot\&\bot))}}{\vdash\bot\lhd(\top\oplus\top),\bot\lhd(\top\oplus\top),\top\oplus(\bot\&\bot)}$$

Figure 3-4: Proof Denoting a Boolean Stack

$$
\mathsf{P_{ana}}\ \dfrac{\dfrac{p_{\mathsf{write}} \qquad \mathsf{P^!_{der}}\ \dfrac{\mathsf{P^!_{con}}\ \dfrac{\mathsf{P_{id}}\ \dfrac{}{\vdash !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot))}}{\vdash !(\bot \lhd (\top \oplus \top)), !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot))}}{\vdash \bot \lhd (\top \oplus \top), !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot))}}{\vdash ((\bot \& \bot) \lhd \top) \& (\bot \lhd (\top \oplus \top)), !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot))}}{\vdash !(((\bot \& \bot) \lhd \top) \& (\bot \lhd (\top \oplus \top))), ?(\top \oslash (\bot \& \bot))}
$$

where p_{write} is

$$
\dfrac{p^1_{\mathsf{write}} \qquad p^2_{\mathsf{write}}}{\vdash (\bot \& \bot) \lhd \top, !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot))}
$$

and p^i_{write} is:

$$
\begin{array}{rc}
\mathsf{P_{id}} & \hline \\
 & \vdash !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash \top, !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot)) \\ \hline
\mathsf{P}_{\oplus i} & \vdash \top \oplus \top, !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash (\top \oplus \top) \oslash !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash ((\top \oplus \top) \oslash !(\bot \lhd (\top \oplus \top))) \parr ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash \bot, ((\top \oplus \top) \oslash !(\bot \lhd (\top \oplus \top))) \parr ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash \bot, (\top \oplus \top), !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash \bot \lhd (\top \oplus \top), !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash \top \oslash !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash (\top \oslash !(\bot \lhd (\top \oplus \top))) \parr ?(\top \oslash (\bot \& \bot)) \\ \hline
 & \vdash \bot \lhd \top, !(\bot \lhd (\top \oplus \top)), ?(\top \oslash (\bot \& \bot))
\end{array}
$$

Figure 3-5: Proof rules for LLP — extends Figure 2-14

$$!\,\frac{\vdash \Gamma^-, N}{\vdash \Gamma^-, !N}\,\mathsf{reuse}(\Gamma^-)$$

$$?d\,\frac{\vdash \Gamma, P}{\vdash \Gamma, ?P} \qquad ?c\,\frac{\vdash \Gamma, N, N}{\vdash \Gamma, N}\,\mathsf{reuse}(N) \qquad ?w\,\frac{\vdash \Gamma}{\vdash \Gamma, N}\,\mathsf{reuse}(N)$$

reusable. $\mathsf{reuse}(\Gamma^-)$ holds if all formulas in Γ^- are reusable. The additional rules of LLP are given in Figure 3-5.

We next extend the embedding in Section 2.6 to full LLP. The LLP exponential is translated to a combination of the corresponding WS exponential and a lift. We set $i(!N) = (\{*\}, \_ \mapsto !\&_{j\in|i(N)|}(\bot \lhd i(N)_j))$ and $i(?P) = (\{*\}, \_ \mapsto ? \bigoplus_{j\in|i(P)|}(\top \oslash i(P)_j))$.

First, we show that each WS! formula in the translation of a reusable LLP formula is equivalent to one with a leading exponential.

Proposition 3.4.2 *Suppose N is reusable. Then for any x in $|i(N)|$, there is a formula Q with proofs $p \vdash !Q^\perp, i(N)_x$ and $p' \vdash i(N)_x^\perp, ?Q$ where $[\![p]\!]$ and $[\![p']\!]$ are inverses.*

Proof We proceed by induction on N. If $N = \bot$ then $i(N)_x = \mathbf{0}$ and the corresponding proofs apply:

$$\frac{}{\vdash \mathbf{1}, ?\mathbf{0}} \qquad \frac{\dfrac{}{\vdash \mathbf{1}, !\mathbf{1}, \mathbf{0}}}{\vdash !\mathbf{1}, \mathbf{0}}$$

If $N = \top$ then $i(N)_x = \top$ and the corresponding proofs apply:

$$\frac{\dfrac{\dfrac{}{\vdash \top}}{\vdash \top, ?\top}}{\vdash ?\top}}{\vdash \bot, ?\top} \qquad \frac{\dfrac{\dfrac{}{\vdash \top}}{\vdash \bot, \top}}{\vdash \bot, !\bot, \top}}{\vdash !\bot, \top}$$

If $N = M \parr L$ then M and L are reusable, and $i(N)_x = i(M)_y \parr i(L)_z$ for some $y \in |i(M)|$ and $z \in |i(L)|$. By induction, there are formulas Q and P with $i(M)_y \cong ?P$ and $i(L)_z \cong ?Q$. We then use the isomorphism $p_1 : !\,M \otimes !\,N \cong !(M\&N) : p_2$ defined in Proposition 3.4.2.

$$\mathsf{P}_{\mathsf{cut}}(p_2)\,\frac{\mathsf{P}^{\mathsf{T}}_{\parr}\,\dfrac{\mathsf{P}_{\mathsf{mul}\otimes}\,\dfrac{\vdash i(M)_y^\perp, ?P \qquad \vdash i(N)_z^\perp, ?Q}{\vdash i(M)_y^\perp \otimes i(N)_z^\perp, ?P, ?Q}}{\vdash i(M)_y^\perp \otimes i(N)_z^\perp, ?P \parr ?Q}}{\vdash i(M)_y^\perp \otimes i(N)_z^\perp, ?(P \oplus Q)} \qquad \mathsf{P}_{\mathsf{cut}}(p_1)\,\frac{\mathsf{P}^{\mathsf{T}}_{\parr}\,\dfrac{\mathsf{P}_{\mathsf{mul}\otimes}\,\dfrac{\vdash !P^\perp, i(M)_y \qquad \vdash !Q^\perp, i(N)_z}{\vdash !P^\perp \otimes !Q^\perp, i(M)_y, i(L)_z}}{\vdash !P^\perp \otimes !Q^\perp, i(M)_y \parr i(L)_z}}{\vdash !(P^\perp \& Q^\perp), i(M)_y \parr i(L)_z}$$

Suppose $N = M\&L$ and $x \in |i(N\&M)| = |i(M)| \uplus |i(N)|$. Then M and L are reusable. Suppose that $x = \mathsf{in}_1(y)$ for $y \in |i(M)|$. Then $i(M\&L)_x = i(M)_y$. By induction, there exists Q and proofs $\vdash i(M)^{\perp}_y, Q$ and $\vdash Q^{\perp}, i(M)_y$ as required. The case when $x = \mathsf{in}_2(y)$ is similar.

If $N = ?P$ then we can take $Q = \bigoplus_{j \in |i(P)|}(\top \oslash i(P)_j)$ and use the $\mathsf{P_{id}}$ rule in each direction. ∎

Next, we must extend Proposition 2.6.1 with the exponential case, to extend the translation of the $\top$ rule of MALLP. $\mathsf{P}^{\mathsf{T}}_{!N,*}$ is defined as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \top}}{\vdash \perp, \top}(\mathsf{P}^{-}_{\perp})^*}{\vdash \perp, !\&_{j\in|i(N)|}(\perp \lhd i(N)_j), \Delta^-, \top}}{\vdash \perp, i(N)_j, !\&_{j\in|i(N)|}(\perp \lhd i(N)_j), \Delta^-, \top}\mathsf{P}^{+}_{\mathsf{wk}}}{\vdash \perp \lhd i(N)_j, !\&_{j\in|i(N)|}(\perp \lhd i(N)_j), \Delta^-, \top} \quad \vdots}{\vdash \&_{j\in|i(N)|}(\perp \lhd i(N)_j), !\&_{j\in|i(N)|}(\perp \lhd i(N)_j), \Delta^-, \top}}{\vdash !\&_{j\in|i(N)|}(\perp \lhd i(N)_j), \Delta^-, \top}$$

We next show how the additional LLP proof rules are translated.

- The ! rule: Let $\Gamma^- = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$. For each $j \in |i(N)|$, $i(q, \overrightarrow{x_i}, j) \vdash \perp, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_j$. We then perform the following derivation r_j:

$$\mathsf{P_{sym}}\dfrac{\dfrac{\dfrac{}{i(q, \overrightarrow{x_i}, n) \vdash \perp, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_j}}{\vdash \perp, i(N)_j, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}{\vdash \perp \lhd i(N)_j, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}$$

  We perform this construction for each j, and using $\mathsf{P_{\&}}$ we obtain

$$r \vdash \&_{j\in|i(N)|}(\perp \lhd i(N)_j), i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$$

  We then set $i(p, \overrightarrow{x_i}) = (*, q)$ where q is:

$$\dfrac{\dfrac{\dfrac{}{r \vdash \&_{j\in|i(N)|}(\perp \lhd i(N)_j), i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}{\vdash \&_{j\in|i(N)|}(\perp \lhd i(N)_j), ?Q_1, \ldots, ?Q_n}}{\vdash !\&_{j\in|i(N)|}(\perp \lhd i(N)_j), ?Q_1, \ldots, ?Q_n}\mathsf{P_{prom}}$$
$$\dfrac{}{\vdash !\&_{j\in|i(N)|}(\perp \lhd i(N)_j), i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}$$

  Here we use the proofs in Proposition 3.4.2 that show that $i(N_i)_{x_i}$ is isomorphic to $?Q_i$.

- The $?d$ rule, with $p = ?d(q)$: Let $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$. Then $i(q, \overrightarrow{x_i}) = (y, q)$ where $q \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$. Then $i(p, \overrightarrow{x_i})$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{q \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}{\vdash \top, i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}{\vdash \top \oslash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}{\mathsf{P}_{\oplus y} \dfrac{}{\vdash \bigoplus_{j \in |i(P)|} \top \oslash i(P)_j, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}}{\mathsf{P}^{?}_{\mathsf{der}} \dfrac{}{\vdash ?(\bigoplus_{j \in |i(P)|} \top \oslash i(P)_j), i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}}}}{\dfrac{\vdash \bot, i(N_1)_{x_1} \parr \ldots \parr i(N_n)_{x_n} \parr ?(\bigoplus_{j \in |i(P)|} \top \oslash i(P)_j)}{\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, ?(\bigoplus_{j \in |i(P)|} \top \oslash i(P)_j)}}$$

- The $?c$ rule, with $p = ?c(q)$:

  If $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$ and $x \in |i(N)|$ then $i(q, \overrightarrow{x_i}, x, x)$ is a proof of $\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_x, i(N)_x$. We can apply Proposition 3.4.2 and use $?$-contraction in WS to yield a proof q' of $\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_x$ and we set $i(p, \overrightarrow{x_i}, x) = q'$.

  If $\Gamma = N_1, \ldots, N_i, P, N_{i+1}, \ldots, N_n$ and $x_i \in |i(N_i)|$ and $x \in |i(N)|$ then $i(q, \overrightarrow{x_i}, x, x) = (y, q')$ where $q' \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_x, i(N)_x$. We can apply Proposition 3.4.2 and use $?$-contraction in WS to yield a proof q'' of

  $$\vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_x$$

  and we set $i(p, \overrightarrow{x_i}, x) = (y, q'')$.

- The $?w$ rule, with $p = ?w(q)$:

  If $\Gamma = N_1, \ldots, N_n$ and $x_i \in |i(N_i)|$ and $x \in |i(N)|$ then $i(q, \overrightarrow{x_i}) \vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$. We can apply $\mathsf{P}^{+}_{\mathsf{wk}}$ in WS to yield a proof q' of $\vdash \bot, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_x$ and we set $i(p, \overrightarrow{x_i}, x) = q'$.

  If $\Gamma = N_1, \ldots, , N_i, P, N_{i+1}, \ldots N_n$ and $x_i \in |i(N_i)|$ and $x \in |i(N)|$ then $i(q, \overrightarrow{x_i}) = (y, q')$ where $q' \vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}$. We can use $\mathsf{P}^{+}_{\mathsf{wk}}$ to yield a proof q'' of $\vdash i(P)_y, i(N_1)_{x_1}, \ldots, i(N_n)_{x_n}, i(N)_x$ and we set $i(p, \overrightarrow{x_i}, x) = (y, q'')$.

We can hence interpret proofs in LLP as (families of) proofs in $\mathsf{WS!}$.

3.5 Semantics of WS!

Definition A WS!-category is a WS-category with a coalgebraic exponential comonoid.

We will give semantics of $\mathsf{WS!}$ in any $\mathsf{WS!}$-category.

Proposition 3.5.1 *$\mathcal{W}$ and $\mathcal{G}$ are WS!-categories.*

Proof Follows from Propositions 3.3.4 and 2.3.3. ∎

3.5.1 Semantics of Sequents

We extend the interpretation of formulas and sequents in Section 2.3.3 to WS! by setting $[\![!N]\!] = ![\![N]\!]$ and $[\![?P]\!] = ![\![P]\!]$.

3.5.2 Semantics of Proofs

The interpretations of the new proof rules are given in Figure 3-6. Once again, proofs of $\vdash N, \Gamma$ are interpreted by arrows $I \to [\![N, \Gamma]\!]$ and proofs of $\vdash P, \Gamma$ by arrows $[\![P, \Gamma]\!] \to \bot$.

Figure 3-6: Semantics for WS! — extends Figures 2-4, 2-5, 2-6

Core rules:

$$\mathsf{P}_! \ \frac{\sigma : [\![\vdash N, !N, \Gamma]\!]}{[\![\Gamma]\!]^-(\alpha^{-1}) \circ \sigma : [\![\vdash !N, \Gamma]\!]} \qquad \mathsf{P}_? \ \frac{\sigma : [\![\vdash P, ?P, \Gamma]\!]}{\sigma \circ [\![\Gamma]\!]^+(\alpha) : [\![\vdash ?P, \Gamma]\!]}$$

Other rules:

$$\mathsf{P}_{\mathsf{ana}} \ \frac{\sigma : [\![\vdash M, P^\perp, P]\!]}{\Lambda_I(\mathbb{(} \Lambda_I^{-1}(\sigma) \mathbb{)}) : [\![\vdash !M, P]\!]}$$

$$\frac{\sigma : [\![\vdash P, \Gamma, !M, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{der} \multimap \mathsf{id}) : [\![\vdash P, \Gamma, M, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash !M, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{der}) \circ \sigma : [\![\vdash M, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash P, \Gamma, !M, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+((\mathsf{d} \multimap \mathsf{id}) \circ \mathsf{pasc}_\multimap^{-1}) : [\![\vdash P, \Gamma, !M, !M, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash N, \Gamma, !M, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{id} \oslash \mathsf{der}) \circ \sigma : [\![\vdash N, \Gamma, M, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash N, \Gamma, !M, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{pasc} \circ (\mathsf{id} \oslash \mathsf{d})) \circ \sigma : [\![\vdash N, \Gamma, !M, !M, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash !M, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{con}) \circ \sigma : [\![\vdash !M, !M, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash M, \Gamma, ?P, ?P, \Delta]\!]}{[\![\Delta]\!]^-((\mathsf{d} \multimap \mathsf{id}) \circ \mathsf{pasc}_\multimap^{-1}) \circ \sigma : [\![\vdash M, \Gamma, ?P, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash ?P, ?P, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{con}) : [\![\vdash ?P, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash Q, \Gamma, ?P, ?P, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{pasc} \circ (\mathsf{id} \oslash \mathsf{d})) : [\![\vdash Q, \Gamma, ?P, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash P, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{der}) : [\![\vdash ?P, \Delta]\!]}$$

$$\frac{\sigma : [\![\vdash Q, \Gamma, P, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^+(\mathsf{id} \oslash \mathsf{der}) : [\![\vdash Q, \Gamma, ?P, \Delta]\!]} \qquad \frac{\sigma : [\![\vdash M, \Gamma, P, \Delta]\!]}{[\![\Delta]\!]^-(\mathsf{der} \multimap \mathsf{id}) \circ \sigma : [\![\vdash M, \Gamma, ?P, \Delta]\!]}$$

3.6 Full Completeness

We will next extend the full completeness result of WS to WS!. We restrict our attention to the concrete games model. We first show that each bounded winning strategy on a type object is the denotation of a unique analytic proof.

Definition Let σ be a winning strategy on a (win-)game G. We say σ is *bounded* if there is $n \in \mathbb{N}$ such that $\forall s \in \sigma$, $|s| \leqslant n$.

Note that a bounded winning strategy on a win-game is precisely a bounded total strategy on that game.

Interpretations of proofs in WS are bounded. Winning strategies on games involving the ! operator are typically not bounded, e.g. there is no bounded winning strategy on $!(\bot \lhd \top)$. However, strategies on e.g. $!\mathbf{B} \multimap \mathbf{B}$ that interrogate their arguments only a finite number of times are bounded.

3.6.1 Reification of Bounded Strategies

Let $\sigma : [\![\vdash \Gamma]\!]$ be a bounded winning strategy on the denotation of a $\mathsf{WS!}$ sequent. We define $\mathsf{reify}(\sigma)$ as an analytic proof of $\vdash \Gamma$. This is defined in Figure 3-7.

Figure 3-7: Reification of Strategies for $\mathsf{WS!}$ — extends Figure 2-7

$$\begin{array}{rcl} \mathsf{reify}_{!N,\Gamma}(\sigma) & = & \mathsf{P}_!(\mathsf{reify}_{N,!N,\Gamma}([\![\Gamma]\!]^-(\alpha) \circ \sigma)) \\ \mathsf{reify}_{?P,\Gamma}(\sigma) & = & \mathsf{P}_?(\mathsf{reify}_{P,?P,\Gamma}(\sigma \circ [\![\Gamma]\!]^+(\alpha^{-1}))) \end{array}$$

We can use the measures defined in Proposition 2.4.3 to show that reify terminates. However, for the first measure, we must use $\mathsf{depth}(\sigma)$ rather than the size of the sequent.

- If $\Gamma = !N, \Gamma'$ then the in the inductive call of reify_Γ, the first measure $\mathsf{depth}(\sigma)$ stays the same. If $N = \bot$ then the second measure $\mathsf{tl}(\Gamma)$ decreases. If $N \neq \bot$ then the second measure stays the same, and the third measure $\mathsf{hd}(\Gamma)$ decreases.
- If $\Gamma = ?P, \Gamma'$ then the in the inductive call of reify_Γ, the first measure $\mathsf{depth}(\sigma)$ stays the same. If $P = \top$ then the second measure $\mathsf{tl}(\Gamma)$ decreases. If $N \neq \top$ then the second measure stays the same, and the third measure $\mathsf{hd}(\Gamma)$ decreases.

3.6.2 Soundness and Uniqueness

We can show that $\mathsf{reify}(\sigma)$ is the unique analytic proof p with $[\![p]\!] = \sigma$.

Proposition 3.6.1 *For any bounded winning strategy* $\sigma : [\![\Gamma]\!]$, $[\![\mathsf{reify}(\sigma)]\!] = \sigma$.

Proof We proceed by induction on the termination measure. In the case when the head formula of Γ is not an exponential, we proceed as in Proposition 2.4.4 using the fact that $\mathcal{W}$ is a complete WS-category. In the other cases:

- If $\Gamma = !\,N, \Gamma'$ then $[\![\mathsf{reify}(\sigma)]\!] = [\![\mathsf{P}_!(\mathsf{reify}([\![\Gamma]\!]^-(\alpha)\circ\sigma))]\!] = [\![\Gamma]\!]^-(\alpha^{-1})\circ[\![\mathsf{reify}([\![\Gamma]\!]^-(\alpha)\circ\sigma)]\!] = [\![\Gamma]\!]^-(\alpha^{-1}) \circ [\![\Gamma]\!]^-(\alpha) \circ \sigma = \sigma$ as required.
- If $\Gamma = ?\,P, \Gamma'$ then $[\![\mathsf{reify}(\sigma)]\!] = [\![\mathsf{P}_?(\mathsf{reify}(\sigma\circ[\![\Gamma]\!]^+(\alpha^{-1})))]\!] = [\![\mathsf{reify}(\sigma\circ[\![\Gamma]\!]^+(\alpha^{-1}))]\!]\circ [\![\Gamma]\!]^+(\alpha) = \sigma \circ [\![\Gamma]\!]^+(\alpha^{-1}) \circ [\![\Gamma]\!]^+(\alpha) = \sigma$ as required. ∎

Proposition 3.6.2 *For any analytic proof p,* $\mathsf{reify}([\![p]\!]) = p$.

Proof We proceed by induction on p. In the cases when p uses one of the core rules of WS, we can proceed as in Proposition 2.4.5 since $\mathcal{W}$ is a complete WS-category. In the other cases:

- If $p = \mathsf{P}_!(q)$ then $\mathsf{reify}([\![p]\!]) = \mathsf{reify}([\![\Gamma]\!]^-(\alpha^{-1})\circ[\![q]\!]) = \mathsf{P}_!(\mathsf{reify}([\![\Gamma]\!]^-(\alpha)\circ[\![\Gamma]\!]^-(\alpha^{-1})\circ [\![q]\!])) = \mathsf{P}_!(\mathsf{reify}([\![q]\!])) = \mathsf{P}_!(q) = p$ as required.
- If $p = \mathsf{P}_?(q)$ then $\mathsf{reify}([\![p]\!]) = \mathsf{reify}([\![q]\!] \circ [\![\Gamma]\!]^+(\alpha)) = \mathsf{P}_?(\mathsf{reify}([\![q]\!] \circ [\![\Gamma]\!]^+(\alpha) \circ [\![\Gamma]\!]^+(\alpha^{-1}))) = \mathsf{P}_?(\mathsf{reify}([\![q]\!])) = \mathsf{P}_?(q) = p$ as required. ∎

Remark Note that the only place we have used the concrete games structure above is in the termination argument, using the depth of a strategy. The reification procedure can be defined in any WS!-category that is a complete WS-category. If it can be shown to terminate in that category, the above propositions show that $\mathsf{reify}(\sigma)$ is the unique analytic proof p with $[\![p]\!] = \sigma$.

3.7 Proof Normalisation

We have seen that any bounded winning strategy is the denotation of a unique analytic proof of WS!. We cannot use this to normalise proofs to their analytic form as for WS, because proofs in WS! do not necessarily denote bounded strategies. We will next show that our reification procedure can be extended to winning strategies that may be unbounded, provided the resulting analytic proofs are allowed to be *infinitary* — that is, proofs using the core rules that may be infinitely deep. More precisely, we will show that *total* strategies on a type object correspond precisely to the infinitary analytic proofs. Thus we can normalise any proof of WS! to its infinitary normal form, by taking its semantics and then constructing the corresponding infinitary analytic proof. Two proofs in WS! are semantically equivalent if and only if they have the same normal form as an infinitary analytic proof.

Remark Systems such as Intuitionistic Linear Logic have exponentials, and cut elimination theorems where the normal form is still a finite proof. This raises the question: why are normal forms of WS! proofs infinitary? We give an informal answer.

Proofs in Intuitionistic Linear Logic can be given semantics as innocent strategies [10], and any innocent strategy $\sigma : !N$ must behave the same in each thread (equationally, $\sigma = (\mathsf{der} \circ \sigma)^\dagger$). Thus, if there are finitely many innocent strategies on N, there are finitely many innocent strategies on $!N$. Proofs in WS! represent history-sensitive strategies, which may behave differently in each of the ω copies of N in $!N$ (where behaviour in each thread can be dependent on behaviour in other threads). For example, a strategy on $!\mathbf{B}$ represents an arbitrary infinite stream of Booleans, which may not even be computable. Thus it will not be expressible by any standard notion of finite analytic proof.

3.7.1 Infinitary Analytic Proofs

We next give formal definitions of analytic proofs that may be infinitary.

Infinitary Proofs as a Final Coalgebra

Let L be a set. Let $\mathcal{T}_L$ denote the final coalgebra of the functor $X \mapsto L \times X^*$ in **Set**. The inhabitants of T_L are L-labelled trees of potentially infinite depth. We let $\alpha : \mathcal{T}_L \to L \times \mathcal{T}_L^*$ describe the arrow part of this final coalgebra: this maps a tree to its label and sequence of subtrees. Given a natural number n, we define a function $N_n : \mathcal{T}_L \to \mathcal{P}(L \times \mathcal{T}_L^*)$, by induction:

- $N_0(T) = \emptyset$
- $N_{n+1}(T) = \{\alpha(T)\} \uplus \bigcup\{N_n(T') : T' \in \pi_2(\alpha(T))\}$

We define the set of nodes $N(T)$ to be $\{N_n(T) : n \in \mathbb{N}\}$.

Let Prf be the set of (names of) proof rules of WS! and Seq the set of sequents of WS!.

Definition An *infinitary analytic proof* of WS! is an infinitary proof using only the core rules of WS!. Formally, this is an element T of $\mathcal{I} = \mathcal{T}_{\mathsf{Prf} \times \mathsf{Seq}}$ such that for each node $((\mathsf{P}_x, \vdash \Gamma), c) \in N(T)$ we have $|c| = \mathsf{ar}(\mathsf{P}_x)$ and if $(\pi_2 \circ \pi_1 \circ \alpha)(c_i) = \vdash \Gamma_i$ then the following is a valid core rule of WS!:

$$\mathsf{P}_x \, \frac{\vdash \Gamma_1 \qquad \dots \qquad \vdash \Gamma_{|c|}}{\vdash \Gamma}$$

We let $\mathcal{I}_\Gamma$ denote the set of infinitary analytic proofs of $\vdash \Gamma$.

Remark Alternatively, we could formulate the core proof rules as an endofunctor on $\mathsf{Set}^{\mathsf{Seq}}$. The analytic proofs then represent the initial algebra of this functor, and the infinitary analytic proofs represent the final coalgebra. We chose the above formulation for simplicity. However, the generated coinductive principle needs refinement.

Let $\{A_\Gamma : \Gamma \in \mathsf{Seq}\}$ be a family of sets indexed by sequents. We next show that we can construct a family of maps $A_\Gamma \to \mathcal{I}_\Gamma$ by giving, for each Γ and $a \in A_\Gamma$, a proof rule that concludes $\vdash \Gamma$ from $\vdash \Gamma_1, \ldots, \vdash \Gamma_i$ and for each i an element $a_i \in A_{\Gamma_i}$. To see this, let $f : \sum_{\Gamma \in \mathsf{Seq}} A_\Gamma \to (\mathsf{Prf} \times \mathsf{Seq}) \times (\sum_{\Gamma \in \mathsf{Seq}} A_\Gamma)^*$ be a function such that for each a, $f(\mathsf{in}_\Gamma(a)) = ((\mathsf{P}_x, \vdash \Gamma), \mathsf{in}_{\Gamma_1}(a_1) \ldots \mathsf{in}_{\Gamma_n}(a_n))$ where P_x has arity n and the following is a valid core rule in $\mathsf{WS!}$:

$$\mathsf{P}_x \, \frac{\vdash \Gamma_1 \qquad \ldots \qquad \vdash \Gamma_n}{\vdash \Gamma}$$

Then we can use the final coalgebraic property of $\mathcal{I}$ to construct a map $\sum_{\Gamma \in \mathsf{Seq}} A_\Gamma \to \mathcal{I}$:

$$\begin{array}{ccc} \displaystyle\sum_{\Gamma \in \mathsf{Seq}} A_\Gamma & \xrightarrow{f} & (\mathsf{Prf} \times \mathsf{Seq}) \times (\displaystyle\sum_{\Gamma \in \mathsf{Seq}} A_\Gamma)^* \\ {\scriptstyle (\!|\, f \,|\!)} \downarrow & & \downarrow {\scriptstyle \mathsf{id} \times (\!|\, f \,|\!)^*} \\ \mathcal{I} & \xrightarrow[\alpha]{} & (\mathsf{Prf} \times \mathsf{Seq}) \times \mathcal{I}^* \end{array}$$

We need to check that for all $\mathsf{in}_\Gamma(a)$, $(\!|\, f \,|\!)(\mathsf{in}_\Gamma(a)) \in \mathcal{I}_\Gamma$. That is, for any n and any $\mathsf{in}_\Gamma(a)$, each element of $N_n((\!|\, f \,|\!)(\mathsf{in}_\Gamma(a)))$ specifies a valid instance of a proof rule of $\mathsf{WS!}$ as described above. We proceed by induction on n. If $n = 0$ this is vacuously true. If $n = m + 1$ then each node in $N_n((\!|\, f \,|\!)(\mathsf{in}_\Gamma(a)))$ is either in $N_m((\!|\, f \,|\!)(\mathsf{in}_\Gamma(a)))$ or is $\alpha((\!|\, f \,|\!)(\mathsf{in}_\Gamma(a)))$. In the former case we are done (by induction). In the latter case, we know that $\alpha((\!|\, f \,|\!)(\mathsf{in}_\Gamma(a))) = (\mathsf{id} \times (\!|\, f \,|\!)^*)(f(\mathsf{in}_\Gamma(a)))$ using the diagram above. By requirement, this is $(\mathsf{id} \times (\!|\, f \,|\!)^*)((\mathsf{P}_x, \vdash \Gamma), \mathsf{in}_{\Gamma_1}(a_1) \ldots \mathsf{in}_{\Gamma_n}(a_n))$ where

$$\mathsf{P}_x \, \frac{\vdash \Gamma_1 \qquad \ldots \qquad \vdash \Gamma_n}{\vdash \Gamma}$$

is a valid proof rule. Thus this node is of the form

$$((\mathsf{P}_x, \vdash \Gamma), (\!|\, f \,|\!)(\mathsf{in}_{\Gamma_1}(a_1)) \ldots (\!|\, f \,|\!)(\mathsf{in}_{\Gamma_n}(a_n))).$$

Since $\alpha((\!|\, f \,|\!)(\mathsf{in}_{\Gamma_i}(a_i))) = ((\mathsf{id} \times (\!|\, f \,|\!)^*) \circ f)(\mathsf{in}_{\Gamma_i}(a_i))$ and $f(\mathsf{in}_{\Gamma_i}(a_i))$ is of the form $((\_, \vdash \Gamma_i), \_)$, so is $(\!|\, f \,|\!)(\mathsf{in}_{\Gamma_i}(a_i))$. Thus $((\mathsf{P}_x, \vdash \Gamma), (\!|\, f \,|\!)(\mathsf{in}_{\Gamma_1}(a_1)) \ldots (\!|\, f \,|\!)(\mathsf{in}_{\Gamma_n}(a_n)))$ does have the structure of a valid proof rule of $\mathsf{WS!}$.

Infinitary Proofs as a Limit of Paraproofs

We can consider an alternative approach for presenting our infinitary analytic proofs. We consider partial proofs, that may give up in the style of [29].

Definition An *analytic paraproof* of $\mathsf{WS!}$ is a proof made up of the core proof rules of $\mathsf{WS!}$, together with a diamon rule P_ϵ that can prove any sequent.

Note that each analytic proof is also a analytic paraproof. Let $\mathcal{C}_\Gamma$ represent the set of analytic paraproofs of $\vdash \Gamma$. We can introduce an ordering $\sqsubseteq$ on this set, generated from the least congruence with P_ϵ as a bottom element. We can take the completion of $\mathcal{C}_\Gamma$ with respect to ω-chains generating an algebraic cpo $\mathcal{D}_\Gamma$. The maximal elements in this domain are precisely the infinitary analytic proofs $\mathcal{I}_\Gamma$, and the compact elements are the analytic paraproofs $\mathcal{C}_\Gamma$.

3.7.2 Semantics of Infinitary Analytic Proofs

We next describe semantics of infinitary analytic proofs via the semantics of analytic paraproofs.

Semantics of Analytic Paraproofs

We can interpret analytic paraproofs as partial strategies. We interpret paraproofs in $\mathcal{G}$, the category of (win) games and strategies. For the rules other than P_ϵ, we use the fact that $\mathcal{G}$ is a WS!-category. We interpret P_ϵ as the strategy $\{\epsilon\}$ where ϵ denotes the empty play on any game. We can hence interpret an analytic paraproof of $\vdash \Gamma$ as a strategy on $[\![\vdash \Gamma]\!]$.

The category $\mathcal{G}$ is cpo-enriched, with $\sigma \sqsubseteq \tau$ if $\sigma \subseteq \tau$ as a set of plays. The bottom element is $\{\epsilon\}$. Composition, pairing and currying are continuous maps of hom sets.

Proposition 3.7.1 *If p and q are analytic paraproofs of $\vdash \Gamma$ and $p \sqsubseteq q$ then $[\![p]\!] \sqsubseteq [\![q]\!]$.*

Proof A simple induction on q, using the fact that composition, pairing and currying are monotonic operations. Note that $[\![-]\!]$ is also strict, as $[\![\mathsf{P}_\epsilon]\!] = \{\epsilon\}$. ∎

Semantics of Infinitary Analytic Proofs

Both $\mathcal{D}_\Gamma$ and hom sets of $\mathcal{G}$ are algebraic domains: each element is the limit of its compact (finite) approximants. Our monotonic map $\mathcal{C}_\Gamma \to [\![\vdash \Gamma]\!]$ thus extends uniquely to a continuous map $\mathcal{D}_\Gamma \to [\![\vdash \Gamma]\!]$. By construction this agrees with the semantics given above for analytic paraproofs in $\mathcal{D}_\Gamma$. Given any infinitary analytic proof p if $p\downarrow$ is the set of analytic paraproofs less than p then $[\![p]\!] = \bigsqcup [\![p\downarrow]\!]$ using the cpo structure in $\mathcal{G}$.

We can show that this really does capture the intended semantics of infinitary analytic proofs.

Proposition 3.7.2 *The equations for the semantics of analytic proofs given in Figures 2-4 and 3-6 hold for infinitary analytic proofs.*

Proof We use the fact that the constructs used in the semantics of the core proof rules are continuous. We proceed by case analysis on the proof rule.

We just give an example. In the case of $\mathsf{P}_\otimes$, note that $[\![\mathsf{P}_\otimes(p,q)]\!] = \bigsqcup\{[\![r]\!] : r \sqsubseteq \mathsf{P}_\otimes(p,q)\} = \bigsqcup\{[\![\mathsf{P}_\otimes(p',q')]\!] : p' \sqsubseteq p \wedge q' \sqsubseteq q\} = \bigsqcup\{[\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle [\![p']\!], [\![q']\!]\rangle : p' \sqsubseteq p \wedge q' \sqsubseteq q\} = [\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle [\![\bigsqcup\{p' : p' \sqsubseteq p\}]\!], [\![\bigsqcup\{q' : q' \sqsubseteq q\}]\!]\rangle = [\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle [\![p]\!], [\![q]\!]\rangle$ as required. All other cases are similar. ∎

We next show that the semantics of an infinitary analytic proof is a total strategy.

Totality

We need to show that given $p \in \mathcal{I}_\Gamma$, $[\![p]\!]$ is a total strategy. Note that this is not true of arbitrary paraproofs in $\mathcal{D}_\Gamma$, nor is it true for infinite derivations in full $\mathsf{WS!}$ (for example, one could repeatedly apply the $\mathsf{P}_{\mathsf{sym}}$ rules forever).

To show this fact, we first introduce some auxiliary notions.

Definition Let $\sigma : N$ be a strategy on a negative game. We say that σ is n-total if whenever $s \in \sigma \wedge |s| \leqslant n \wedge so \in P_N \Rightarrow \exists p.sop \in \sigma$.

It is clear that a strategy is total if and only if it is n-total for each n.

Proposition 3.7.3 *The following facts hold:*

1. *If σ is n-total and τ is an isomorphism then $\tau \circ \sigma$ is n-total.*
2. *If σ is n-total and τ is an isomorphism then $\sigma \circ \tau$ is n-total.*
3. *If $\sigma : A \otimes B \multimap C$ is n-total then $\Lambda(\sigma)$ is n-total.*
4. *If $\sigma : A \to B$ and $\tau : A \to C$ are n-total then $\langle \sigma, \tau\rangle$ is also n-total.*
5. *If $\sigma : A_i \multimap B$ is n-total then $\sigma \circ \pi_i : A_1 \times A_2 \multimap B$ is n-total*
6. *If $\sigma : A \multimap B$ is n-total then $\sigma \multimap \mathsf{id} : (B \multimap o) \multimap (A \multimap o)$ is $(n+2)$-total.*

Proof 1. Suppose $\sigma : A \multimap B$ is n-total and $\tau : B \multimap C$ is an isomorphism. Note that τ induces an isomorphism of plays $f_\tau : P_B \cong P_C$. This extends to an isomorphism of plays $f_{A\multimap\tau} : P_{A\multimap B} \cong P_{A\multimap C}$. If $|s| \leqslant n$, $s \in \sigma \circ \tau$ and $so \in P_{A\multimap C}$ then $f^{-1}_{A\multimap\tau}(so) \in P_{A\multimap B}$. We know that $f^{-1}_{A\multimap\tau}(s) \in \sigma$, $|f^{-1}_{A\multimap\tau}(s)| \leqslant n$ and $|f^{-1}_{A\multimap\tau}(so)|$ extends $|f^{-1}_{A\multimap\tau}(s)|$ by a single O-move. Thus by n-totality of σ, there is a move p such that $f^{-1}_{A\multimap\tau}(so)p \in \sigma$. Then $f_{A\multimap\tau}(f^{-1}_{A\multimap\tau}(so)p) \in \tau$. This is a play that extends so by a single P-move q. Thus $\tau \circ \sigma$ is n-total.

2. Similar to the previous case.
3. Let $s \in \Lambda(\sigma)$ with $so \in P_{A \multimap (B \multimap C)}$ and $|s| \leqslant n$. Then so corresponds to a play $s'o'$ in $P_{A \otimes B \multimap C}$, with $s' \in \sigma$, as the Λ operation renames indices of moves in a bijective fashion. Since σ is n-total, there is a response p' with $s'o'p' \in \sigma$. Then by applying the appropriate relabelling we see that this is corresponds to a play $sop \in \Lambda(\sigma)$, as required.
4. Let $s \in \langle \sigma, \tau \rangle$ with $so \in P_{A \multimap B \times C}$. Each nonempty play in $A \multimap B \times C$ corresponds either to a play in $A \multimap B$ or $A \multimap C$ depending on the first move. Assume wlog so is a play in $A \multimap B$. Then since σ is n-total, so has a response in σ, and hence in $\langle \sigma, \tau \rangle$, as required.
5. Let $s \in \sigma \circ \pi_i$ with $so \in A_1 \times A_2 \multimap B$ and $|s| \leqslant n$. Then s is also a play in $\sigma : A_i \multimap B$ up to retagging, and so a play in $A_i \multimap B$ by the switching condition. Thus by n-totality of σ, σ provides a response p in $A_i \multimap B$. Then $sop \in \sigma$ is also in $\sigma \circ \pi_i$, as required.
6. Let $s \in \sigma \multimap \mathsf{id}$ be such that $|s| \leqslant n + 2$ and $so \in P_{(B \multimap o) \multimap (A \multimap o)}$. If $s = \epsilon$ then o must be the initial move, and we know that $\sigma \multimap \mathsf{id}$ provides a response to this move. Otherwise, one can remove the first two moves of so, to generate a play $s'o'$ in $P_{A \multimap B}$ with $s' \in \sigma, |s'| \leqslant n$. By n-totality of σ, there exists p' with $s'o'p' \in \sigma$. By applying the appropriate relabelling we find a move p such that $sop \in \sigma \multimap \mathsf{id}$, as required. ∎

Proposition 3.7.4 *Given any infinitary analytic proof p of $\vdash \Gamma$, $[\![p]\!]$ is total.*

Proof We show that $[\![p]\!]$ is n-total for each n. We proceed by induction on a compound measure.

- Define $\mathsf{tl}^+(A, \Gamma)$ to be the length of Γ as a list if $A = \top$ or ∞ otherwise.
- Define $\mathsf{hd}^+(A, \Gamma)$ to be $|A|$ if A is positive or ∞ otherwise.
- Define $\mathsf{tl}^-(A, \Gamma)$ to be the length of Γ as a list if $A = \bot$ or ∞ otherwise.
- Define $\mathsf{hd}^-(A, \Gamma)$ to be $|A|$ if A is negative or ∞ otherwise.

We proceed by induction on $f(n, \Gamma) = \langle n, \mathsf{tl}^+(\Gamma), \mathsf{hd}^+(\Gamma), \mathsf{tl}^-(\Gamma), \mathsf{hd}^-(\Gamma) \rangle$ in the lexicographical ordering on $\mathbb{N} \times \mathbb{N} \cup \{\infty\} \times \mathbb{N} \cup \{\infty\} \times \mathbb{N} \cup \{\infty\} \times \mathbb{N} \cup \{\infty\}$. We proceed by case analysis on p.

- If $p = \mathsf{P}_{\mathbf{1}}$ or $\mathsf{P}_{\top}$ then p is a finite proof, hence $[\![p]\!]$ is total by semantics of WS.
- If $p = \mathsf{P}_{\otimes}(p_1, p_2)$ then by Proposition 3.7.2 we see that $[\![\mathsf{P}_{\otimes}(p_1, q_2)]\!] = [\![\Gamma]\!]^-(\mathsf{dec}^{-1}) \circ \mathsf{dist}^{-1}_{-,\Gamma} \circ \langle [\![p_1]\!], [\![p_2]\!] \rangle$. We know by induction that $[\![p_1]\!]$ and $[\![p_2]\!]$ are n-total. The

call to the inductive hypothesis is valid because $f(n,(M_i,\Gamma)) < f(n,(M_1 \otimes M_2,\Gamma))$ — it is smaller in either the fourth or fifth components, and equal in previous components. Thus by by Proposition 3.7.3 $[\![\mathsf{P}_\otimes(p,q)]\!] = [\![p]\!]$ is n-total.

- If $p = \mathsf{P}_{\parr 1}(q)$ then $[\![p]\!] = [\![q]\!] \circ \Delta^+(\mathsf{wk} \circ \mathsf{sym}) = [\![q]\!] \circ \Delta^+(\pi_1 \circ \mathsf{dec} \circ \mathsf{sym}) = [\![q]\!] \circ \Delta^+(\pi_1) \circ \Delta^+(\mathsf{dec} \circ \mathsf{sym}) = [\![q]\!] \circ \pi_1 \circ \mathsf{dist}_{+,\Delta} \circ \Delta^+(\mathsf{dec} \circ \mathsf{sym})$. By induction (smaller in the second or third component), $[\![q]\!]$ is n-total, and so by Proposition 3.7.3 $[\![p]\!]$ is n-total. The case of $p = \mathsf{P}_{\parr 2}(q)$ is similar.
- The remaining cases work in an entirely analogous way. For $\mathsf{P}^+_\bot$ we must use the fact that currying preserves n-totality. For termination:
  - In the case of $\mathsf{P}_\otimes$, $\mathsf{P}_\&$, $\mathsf{P}_!$ the first three measures $(n, \mathsf{tl}^+(\Gamma), \mathsf{hd}^+(\Gamma))$ stay the same and either the fourth measure $\mathsf{tl}^-(\Gamma)$ decreases, or the fourth measure stays the same and the fifth measure $\mathsf{hd}^-(\Gamma)$ decreases.
  - In the case of $\mathsf{P}^{\parr}_\bot$, $\mathsf{P}^{\oslash}_\bot$, $\mathsf{P}^-_\bot$ the first three measures stay the same and the fourth measure decreases.
  - In the cases of $\mathsf{P}^+_\bot$, $\mathsf{P}_\parr$, $\mathsf{P}_\oplus$, $\mathsf{P}_?$ the first measure n stays the same and either the second measure $\mathsf{tl}^+(\Gamma)$ decreases, or the second measure stays the same and the third measure $\mathsf{hd}^+(\Gamma)$ decreases.
  - In the case of $\mathsf{P}^{\otimes}_\top$, $\mathsf{P}^{\lhd}_\top$, $\mathsf{P}^{+}_\top$ the first measure stays the same and the second measure decreases.
  - In the case of $\mathsf{P}^-_\top$, the first measure decreases. In particular, $[\![\mathsf{P}^-_\top(q)]\!] = \mathsf{unit}_{\multimap} \circ ([\![q]\!] \multimap \mathsf{id})$. By induction $[\![q]\!]$ is $(n-2)$-total, and so $[\![q]\!] \multimap \mathsf{id}$ is n-total, and so $[\![p]\!]$ is n-total by Proposition 3.7.3. ∎

Note that there are infinitary analytic proofs that denote strategies that are total, but not winning. For example, there is an infinitary analytic proof of $\vdash \bot, ?(\top \lhd \bot)$ given by $\mathsf{P}^+_\bot(h)$ where h is the infinitary analytic proof of $\vdash ?(\top \lhd \bot)$ given by $h = \mathsf{P}_?(\mathsf{P}_\lhd(\mathsf{P}^{\lhd}_\top(\mathsf{P}^-_\top(\mathsf{P}_\lhd(\mathsf{P}^+_\bot(h))))))$. But there are no winning strategies on this game.

3.7.3 Reification of Total Strategies as Infinitary Analytic Proofs

We next show that any total strategy σ on the denotation of a sequent is the interpretation of a unique infinitary analytic proof $\mathsf{reify}(\sigma)$.

We first define reify for winning strategies. We have seen that we can construct a family of maps $A_\Gamma \to \mathcal{I}_\Gamma$ by giving, for each Γ and $a \in A_\Gamma$, a proof rule that concludes

$\vdash \Gamma$ from $\vdash \Gamma_1, \ldots, \vdash \Gamma_i$ and for each i an element $a_i \in A_{\Gamma_i}$.

$$\begin{array}{ccc} \sum_{\Gamma \in \mathsf{Seq}} A_\Gamma & \xrightarrow{f} & (\mathsf{Prf} \times \mathsf{Seq}) \times (\sum_{\Gamma \in \mathsf{Seq}} A_\Gamma)^* \\ \multimap f \multimap \downarrow & & \downarrow \mathsf{id} \times \multimap f \multimap^* \\ \mathcal{I} & \xrightarrow[\alpha]{} & (\mathsf{Prf} \times \mathsf{Seq}) \times \mathcal{I}^* \end{array}$$

Note that our reification function reify defined in Figure 3-7 is exactly of this shape. In this case $A_\Gamma = \mathsf{Win}_\Gamma$, the set of winning strategies on $[\![\Gamma]\!]$. The function specifies, for each strategy, the root-level proof rule and the derived strategies that are given as input to reify coinductively. In the case that σ is bounded, we have seen that the process terminates and $\mathsf{reify}(\sigma)$ is a finite proof.

In fact, we note that this family of maps are still well defined if A_Γ is the set of *total* strategies on $[\![\vdash \Gamma]\!]$.

Proposition 3.7.5 reify_Γ *is well defined for total strategies on* $[\![\vdash \Gamma]\!]$.

Proof Our reification procedure uses the fact that $\mathcal{W}$ is a complete WS!-category. We cannot construct a category of unbounded games and total strategies, as composition is not well-defined in general. However:

- The composition of a total strategy and an isomorphism is a total strategy.
- The composition of a total strategy and a projection is a total strategy.
- The completeness axioms in Definition 2.4.3 are satisfied:
  - There are no total strategies on $\bot$.
  - The map d sending pairs of total strategies on $(M \multimap \bot, N \multimap \bot)$ to total strategies on $M \times N \multimap \bot$ is an isomorphism.
  - The map $\_ \multimap \bot$ sending total strategies on M to total strategies on $(M \multimap \bot) \multimap \bot$ is an isomorphism.

Thus (looking at each case) we see that reify_Γ is well-defined on total strategies. In particular, the procedure provides, for each total strategy on Γ, a proof rule P_x concluding

Γ from $\vdash \Gamma_1, \ldots, \vdash \Gamma_n$ and total strategies on each $[\![\vdash \Gamma_i]\!]$. We write this map as reif_Γ.

$$\begin{array}{ccc} \sum_{\Gamma \in \mathsf{Seq}} \mathsf{Tot}_\Gamma & \xrightarrow{\mathsf{reif}} & (\mathsf{Prf} \times \mathsf{Seq}) \times (\sum_{\Gamma \in \mathsf{Seq}} \mathsf{Tot}_\Gamma)^* \\ \Big\downarrow{\scriptstyle \mathsf{reify} = (\!|\, \mathsf{reif} \,|\!)} & & \Big\downarrow{\scriptstyle \mathsf{id} \times \mathsf{reify}^*} \\ \mathcal{I} & \xrightarrow[\alpha]{} & (\mathsf{Prf} \times \mathsf{Seq}) \times \mathcal{I}^* \end{array}$$

Thus we can take the anamorphism of this map yielding a map from total strategies on Γ to $\mathcal{I}_\Gamma$, as required. ∎

Propositions 3.6.1 and 3.6.2 ensure that in the case that σ is bounded, $\mathsf{reify}(\sigma)$ is the unique analytic proof whose semantics is σ. We next wish to give the analogous result for our infinitary version of reify.

3.7.4 Soundness and Uniqueness

We can show that given any winning strategy σ, $\mathsf{reify}(\sigma)$ is the unique infinitary analytic proof p such that $[\![\mathsf{reify}(p)]\!] = \sigma$.

For soundness, we first introduce some auxiliary notions.

Definition Let σ and τ be strategies on A. We say that $\sigma =_n \tau$ if a) each play in σ of length at most n is in τ and b) each play in τ of length at most n is in σ.

It is clear that $=_n$ is an equivalence relation, and $\sigma = \tau$ if and only if $\sigma =_n \tau$ for each $n \in \mathbb{N}$.

Proposition 3.7.6 *1. If $\sigma =_n \tau$ and ρ is an isomorphism then $\sigma \circ \rho =_n \tau \circ \rho$.*

2. If $\sigma =_n \tau$ and ρ is an isomorphism then $\rho \circ \sigma =_n \rho \circ \tau$.

3. If $\sigma =_n \tau$ and $\rho =_n \delta$ then $\langle \sigma, \rho \rangle =_n \langle \tau, \delta \rangle$.

4. If $\sigma =_n \tau$ then $\Lambda(\sigma) =_n \Lambda(\tau)$.

5. If $\sigma =_n \tau$ then $\sigma \circ \pi_i =_n \tau \circ \pi_i$.

6. If $\sigma =_n \tau$ then $\sigma \multimap \mathsf{id} =_{n+2} \tau \multimap \mathsf{id}$.

Proof Similar to Proposition 3.7.3, noting that plays in the left (resp. right) hand side of the conclusion equation have corresponding plays in the left (resp. right) hand side of the hypothesis equation. ∎

Proposition 3.7.7 *Given any total strategy σ on $[\![\vdash \Gamma]\!]$, we have $[\![\mathsf{reify}(\sigma)]\!] = \sigma$.*

Proof We show that for each n, $[\![\mathsf{reify}(\sigma)]\!] =_n \sigma$. The structure of the induction follows that of Proposition 3.7.4, lexicographically on $\langle n, \mathsf{tl}^+(\Gamma), \mathsf{hd}^+(\Gamma), \mathsf{tl}^-(\Gamma), \mathsf{hd}^-(\Gamma)\rangle$. In each particular case, the reasoning follows the proofs of Proposition 2.4.4 and 3.6.1 using $=_n$ in the inductive hypothesis rather than $=$, and propagating this to the main equation using Proposition 3.7.6. In the case of $\Gamma = \top, N$ we use the inductive hypothesis with a smaller n, using the final clause in Proposition 3.7.6. ∎

Proposition 3.7.8 *Given any infinitary analytic proof p,* $\mathsf{reify}([\![p]\!]) = p$.

Proof Since $\mathsf{id} = \langle\!\langle\, \alpha \,\rangle\!\rangle$, we know that id is the unique morphism f such that:

$$\begin{array}{ccc} \mathcal{I}_\Gamma & \xrightarrow{\alpha} & (\mathsf{Prf} \times \mathsf{Seq}) \times \mathcal{I}^* \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathsf{id} \times f^*} \\ \mathcal{I}_\Gamma & \xrightarrow{\alpha} & (\mathsf{Prf} \times \mathsf{Seq}) \times \mathcal{I}^* \end{array}$$

Thus to show that $\mathsf{reify} \circ [\![-]\!] = \mathsf{id}$ it is sufficient to show that $\alpha \circ \mathsf{reify} \circ [\![-]\!] = \mathsf{id} \times (\mathsf{reify} \circ [\![-]\!])^* \circ \alpha$, i.e. that for each infinitary analytic proof p we have $\alpha(\mathsf{reify}([\![p]\!])) = (\mathsf{id} \times (\mathsf{reify} \circ [\![-]\!])^*)(\alpha(p))$.

- For binary rules P_x we must show that $\mathsf{reify}([\![\mathsf{P}_x(p_1, p_2)]\!]) = \mathsf{P}_x(\mathsf{reify}([\![p_1]\!]), \mathsf{reify}([\![p_2]\!]))$.
- For unary rules P_x we must show that $\mathsf{reify}([\![\mathsf{P}_x(p)]\!]) = \mathsf{P}_x(\mathsf{reify}([\![p]\!]))$
- For nullary rules P_x we must show that $\mathsf{reify}([\![\mathsf{P}_x]\!]) = \mathsf{P}_x$.

For each proof rule, we have already shown this in the proof of Proposition 2.4.5 or Proposition 3.6.2. Proposition 3.7.2 ensures that the proof applies in this setting. ∎

3.7.5 Full Completeness and Normalisation

We have thus shown:

Theorem 3.7.9 *Each total strategy σ on $[\![\vdash \Gamma]\!]$ is the denotation of a unique infinitary analytic proof* $\mathsf{reify}(\sigma)$.

We hence have a bijection between infinitary analytic proofs of a formula, and total strategies on the denotation of that formula, via the semantics. Since any proof in WS! can be given semantics as a winning strategy, and winning strategies are total, we may reify the semantics of a WS! proof to generate its infinitary normal form $\mathsf{reify}([\![p]\!])$.

Theorem 3.7.10 *For each WS! proof p, there is a unique infinitary analytic proof q such that $[\![p]\!] = [\![q]\!]$.*

Proof Let $q = \mathsf{reify}(\llbracket p \rrbracket)$. Then $\llbracket q \rrbracket = \llbracket \mathsf{reify}(\llbracket p \rrbracket) \rrbracket = \llbracket p \rrbracket$ by Proposition 3.7.7. If q' is an infinitary analytic proof with $\llbracket q' \rrbracket = \llbracket p \rrbracket$ then $\llbracket q' \rrbracket = \llbracket q \rrbracket$ and so $\mathsf{reify}(\llbracket q' \rrbracket) = \mathsf{reify}(\llbracket q \rrbracket)$ and Proposition 3.7.8 ensures that $q' = q$. ∎

While infinitary analytic proofs may denote strategies that are not winning, any infinitary analytic proof generated as a result of the above normalisation denotes a winning strategy. The above result also ensures that proofs p_1 and p_2 in $\mathsf{WS!}$ denote the same strategy if and only if their normal forms (as infinitary analytic proofs) are identical.

3.8 Cut Elimination for Analytic Proofs

We can extend our syntactic cut elimination procedure of Section 2.5.1 to $\mathsf{WS!}$. This maps (finite) analytic proofs of $\vdash A, \Gamma, N^\perp$ and $\vdash N, P$ to an analytic proof of $\vdash A, \Gamma, P$.

3.8.1 Cut Elimination Procedure

In Figure 3-8, we extend the syntactic cut elimination procedure to $\mathsf{WS!}$.

Figure 3-8: Cut Elimination for $\mathsf{WS!}$ — extends Figures 2-12, 2-13, 2-10, 2-11

A	Γ	$\mathsf{cut} :\vdash A, \Gamma, N^\perp \times \vdash N, P$	$\to$	$\vdash A, \Gamma, P$			
$!M$		$\mathsf{cut}(\mathsf{P}_!(y, g))$	$=$	$\mathsf{P}_!(\mathsf{cut}(y, g))$			
$?P$		$\mathsf{cut}(\mathsf{P}_?(y, g))$	$=$	$\mathsf{P}_?(\mathsf{cut}(y, g))$			
Q	Γ	$\mathsf{cut}_2 :\vdash Q, \Gamma, N^\perp \times \vdash Q^\perp, \Gamma^\perp, P$	$\to$	$\vdash N^\perp \parr P$			
$?\_$		$\mathsf{cut}_2(\mathsf{P}_?(y), \mathsf{P}_!(g))$	$=$	$\mathsf{cut}_2(y, g)$			
$\mathsf{wk}_P(\mathsf{P}_!(p))$	$=$	$\mathsf{P}_!(\mathsf{wk}_P(p))$	$\mathsf{wk}_P(\mathsf{P}_?(p))$	$=$	$\mathsf{P}_?(\mathsf{wk}_P(p))$		
$\mathsf{rem}_0(\mathsf{P}_!(p))$	$=$	$\mathsf{P}_!(\mathsf{rem}_0(p))$	$\mathsf{wk}_P(\mathsf{P}_?(p))$	$=$	$\mathsf{P}_?(\mathsf{rem}_0(p))$		

3.8.2 Soundness

We can show that this elimination procedure is sound with respect to any WS!-category.

Proposition 3.8.1 *In any WS!-category, if p_1 is a proof of $\vdash A, \Gamma, N^\perp$ and p_2 is a proof of $\vdash N, R$ then $\llbracket \mathsf{cut}(p_1, p_2) \rrbracket = \llbracket \mathsf{P}_{\mathsf{cut}}(p_1, p_2) \rrbracket$.*

The proof is an easy extension to Propositions 2.5.5, 2.5.4 and 2.5.2. The cases for the new rules follow precisely the same pattern as other cases whose interpretation is an isomorphism, e.g. $\mathsf{P}^{\oslash}_{\perp}$.

This concludes our treatment of the sequoidal exponential. In the next chapter, we will extend $\mathsf{WS!}$ to a first-order logic.

Chapter 4

Atoms and Quantifiers

We have seen that WS! has expressive computational power. But from a logical perspective, it is somewhat limited: any proposition must ultimately be composed of units and connectives. In this chapter we introduce atoms, predicates, quantifiers and equality into our logic. Formulas are interpreted as a family of games indexed by first-order models, and proofs as families of strategies that must behave in a uniform manner.

We next introduce WS1, adding atoms and quantifiers to our logic, significantly increasing its expressive power. Semantically, formulas represent families of games, indexed over models: a (negative) atom α may either be true (in which case it is interpreted by $\mathbf{1}$ and has a total strategy) or false (in which case it is interpreted by $\bot$ and has no winning strategy). A proof of a formula denotes a winning strategy that works regardless of the truth values of the atoms: a family of winning strategies that behave in a *uniform* manner.

The atoms themselves are predicates applied to variables, and formulas are interpreted with respect to a given first-order model. Our logic has first-order quantifiers $\forall$ and $\exists$. In the game denoted $\forall x.N(x)$, Opponent specifies an element a in the model and play proceeds in $N(a)$. Thus, a winning strategy on $\forall x.N(x)$ must provide a winning strategy on $N(a)$ for each a in the model.

We will first show how atoms, equality and quantifiers can be accommodated in the logic WS. The treatment of atoms and equalities are non-standard, chosen so that we can extend the full completeness results of previous chapters. We give motivation for these rules based on their (informal) semantics: formulas as families of games (indexed over first-order structures) and proofs as families of strategies upon them. We show how first-order Intuitionistic Linear Logic can be embedded, and identify formulas which are not provable in Intuitionistic Linear Logic but are provable in WS1, including a medial

rule which Blass noticed has a (uniform) winning strategy but no proof in ILL [14].

We then formalise the semantics of WS1. Proofs in WS1 denote strategies which are *uniform* — the family of strategies behaves (in some sense) in the same manner regardless of the underlying model. We formalise this notion using lax natural transformations: a formula is represented as a functor from the category of first-order structures to a category of games, and proofs are represented as lax natural transformations between these functors. To reuse the semantics constructed in previous chapters, we construct a WS!-category of such functors and lax natural transformations. To interpret the quantifiers we exhibit an adjunction and use standard techniques.

We will then show full completeness: each finitary uniform winning strategy is the denotation of a unique analytic proof. To do this, we must show some strong properties of uniform winning strategies with respect to existential quantification and disjunction. Infinitary uniform total strategies correspond to infinitary analytic proofs, and so once again we can normalise proofs to their unique analytic form.

Finally, we show how the syntactic cut elimination procedure can be extended to WS1.

4.1 The Logic WS1

4.1.1 Syntax and Informal Semantics

Formulas of WS1

Our syntax and semantics are given with respect to a particular first-order language.

Definition A *(polarized) first-order language* consists of:

- A collection of complementary pairs of predicate symbols ϕ (negative) and $\overline{\phi}$ (positive), each with an arity in $\mathbb{N}$ such that $\mathsf{ar}(\phi) = \mathsf{ar}(\overline{\phi})$. This must include the binary symbol $=$ (negative), and we write $\neq$ for its complement
- A collection of function symbols, each with an arity.

We fix a set of variables $\mathcal{V}$. Given a first-order language $\mathcal{L}$, we define the set of *terms* to be the set freely generated by the variables and the function symbols. The formulas of WS1 are defined as follows:

$$\begin{array}{rlllllllllll} M, N := & \mathbf{1} & | & \bot & | & \phi(\overrightarrow{s}) & | & M \otimes N & | & M \oslash N & | & N \lhd P & | \\ & \forall x.N & | & M \& N & | & !N & & & & & & & \\ P, Q := & \mathbf{0} & | & \top & | & \overline{\phi}(\overrightarrow{s}) & | & P \parr Q & | & P \lhd Q & | & P \oslash N & | \\ & \exists x.P & | & P \oplus Q & | & ?P & & & & & & & \end{array}$$

Here s ranges over terms, x over variables, and $\phi(\overrightarrow{s})$ over n-ary predicates ϕ applied to a tuple of terms $\overrightarrow{s} = (s_1, \ldots, s_n)$. Equality and inequality are special cases of atoms.

The involutive negation operation $(\_)^{\perp}$ sends negative formulas to positive ones and *vice versa* by exchanging each atom, unit or connective for its dual — i.e. $\mathbf{1}$ for $\mathbf{0}$, $\perp$ for $\top$, $\phi(\overrightarrow{s})$ for $\overline{\phi}(\overrightarrow{s})$, $\otimes$ for $\parr$, $\oslash$ for $\triangleleft$, $\forall$ for $\exists$, & for $\oplus$ and ! for ?.

Interpretation of Formulas

Definition An $\mathcal{L}$-*structure* is a set $|L|$ together with an interpretation map I_L sending:

- each predicate symbol ϕ (with arity n) to a function $I_L(\phi) : |L|^n \to \{\mathtt{tt}, \mathtt{ff}\}$ such that $I_L(\phi)(\overrightarrow{a}) \neq I_L(\overline{\phi})(\overrightarrow{a})$ for all $\vec{a}$ and $I_L(=)(a, b) = \mathtt{tt}$ if and only if $a = b$
- each function symbol f (with arity n) to a function $I_L(f) : |L|^n \to |L|$.

If $X \subseteq \mathcal{V}$ an $\mathcal{L}$-*model over* X is a pair (L, v) where L is an $\mathcal{L}$-structure and $v : X \to |L|$ a valuation function. If (L, v) is an $\mathcal{L}$-model over X and s a term with free variables in X, we can define inductively $I_{(L,v)}(s) \in |L|$. If $\phi(s_1, \ldots, s_n)$ is an atomic formula we write $(L, v) \models \phi(s_1, \ldots, s_n)$ if $I_L(\phi)(I_{(L,v)}(s_1), \ldots, I_{(L,v)}(s_n)) = \mathtt{tt}$, and say that $\phi(s_1, \ldots, s_n)$ is *satisfied* in (L, v).

Given an $\mathcal{L}$-model (L, v) over $\mathcal{V}$ we interpret positive and negative formulas as games:

- Positive atoms which are satisfied in (L, v) are interpreted as the game $\top$ with a single (Player) move; positive atoms which are not satisfied are interpreted as the game $\mathbf{0}$ with no moves.
- Negative atoms which are satisfied in (L, v) are interpreted as the empty game $\mathbf{1}$, whilst negative atoms which are not satisfied are interpreted as the game $\perp$ with single Opponent move.
- In $\forall x.N(x)$, dialogues are played in $N(a)$ for some value $a \in |L|$ chosen by Opponent.
- In $\exists x.P(x)$, dialogues are played in $P(a)$ for some value $a \in |L|$ chosen by Player.

We see that the game $\overline{\phi}(\overrightarrow{s})$ has a winning strategy if $(L, v) \models \overline{\phi}(\overrightarrow{s})$ and $\phi(\overrightarrow{s})$ has a winning strategy if $(L, v) \models \phi(\overrightarrow{s})$.

Proof Rules

With this interpretation in mind, we can define proof rules for WS1. A sequent of WS1 is of the form $X; \Theta \vdash \Gamma$ where $X \subseteq \mathcal{V}$, Θ is a set of atomic formulas and Γ is a nonempty list of formulas such that $FV(\Theta, \Gamma) \subseteq X$. We must specify X explicitly due

Figure 4-1: Proof rules for WS1 — extends Figure 3-1

Core rules:

$$\mathsf{P}_{\mathsf{at}-}\ \frac{X;\Theta,\overline{\phi}(\overrightarrow{s})\vdash\bot,\Gamma}{X;\Theta\vdash\phi(\overrightarrow{s}),\Gamma} \qquad \mathsf{P}_{\mathsf{at}+}\ \frac{X;\Theta,\overline{\phi}(\overrightarrow{s})\vdash\top,\Gamma}{X;\Theta,\overline{\phi}(\overrightarrow{s})\vdash\overline{\phi}(\overrightarrow{x}),\Gamma}$$

$$\mathsf{P}_{\mathsf{ma}}^{x,y,z}\ \frac{(X;\Theta\vdash\Gamma)[\frac{z}{x},\frac{z}{y}] \qquad X;\Theta,x\neq y\vdash\Gamma}{X;\Theta\vdash\Gamma} \qquad \mathsf{P}_{\neq}\ \frac{}{X;\Theta,x\neq x\vdash\Gamma}$$

$$\mathsf{P}_{\forall}\ \frac{X\uplus\{x\};\Theta\vdash N,\Gamma}{X;\Theta\vdash\forall x.N,\Gamma}\ x\notin FV(\Theta,\Gamma) \qquad \mathsf{P}_{\exists}^{s}\ \frac{X;\Theta\vdash P[s/x],\Gamma}{X;\Theta\vdash\exists x.P,\Gamma}\ FV(s)\subseteq X$$

Other rules:

$$\mathsf{P}_{\forall}^{\mathsf{T}}\ \frac{X;\Theta\vdash\Gamma,\forall x.N,\Delta}{X;\Theta\vdash\Gamma,N[s/x],\Delta}\ FV(s)\subseteq X \qquad \mathsf{P}_{\exists}^{\mathsf{T}}\ \frac{X;\Theta\vdash\Gamma,P[s/x],\Delta}{X;\Theta\vdash\Gamma,\exists x.P,\Delta}\ FV(s)\subseteq X$$

$$\mathsf{P}_{\mathsf{fneq}}\ \frac{X;\Theta,s\neq t\vdash\Gamma}{X;\Theta,f(s)\neq f(t)\vdash\Gamma}$$

to the strength of our correspondence between syntax and semantics. For convenience, we assume all bound variables are distinct, and distinct from any free variables (perhaps achieved using an initial α-conversion). We can interpret each sequent as a family of games, indexed over Θ-satisfying $\mathcal{L}$-models over X.

We assume a notion of capture-avoiding substitution, using the usual notation $N[s/x]$ to mean the formula N with free occurrences of x replaced for s. This can be extended to substitution on the $X;\Theta$ component. The proof rules associated to our new operators are defined in Figure 4-1. We include all of the rules of WS! with the $X;\Theta$ contexts propagated additively (to be precise, we require that the $X;\Theta$ context of the conclusion is the same as in each of the premises).

We next informally describe interpretation of the core rules.

- $\mathsf{P}_{\mathsf{at}-}$: $\phi(\overrightarrow{s}),\Gamma$, is interpreted by $\mathbf{1},\Gamma$ if $(L,v)\models\phi(\overrightarrow{s})$ or $\bot,\Gamma$ if $(L,v)\not\models\phi(\overrightarrow{s})$. In the former case, there are no moves to respond to, so we only need to consider the case when $(L,v)\models\overline{\phi}(\overrightarrow{s})$, which is given by the premise.
- $\mathsf{P}_{\mathsf{at}+}$: For each $\Theta,\overline{\phi}(\overrightarrow{s})$-satisfying $\mathcal{L}$-model (L,v), the premise yields a strategy on $\top,\Gamma(L,v)=\overline{\phi}(\overrightarrow{s}),\Gamma(L,v)$, as required.
- $\mathsf{P}_{\forall}$: To give a strategy on $\forall x.N,\Gamma(L,v)$ for each Θ-satisfying $\mathcal{L}$-model over X (L,v), we must give a strategy on $N,\Gamma(L,v)$ for each choice of x — that is, a family of strategies on the set of Θ-satisfying $\mathcal{L}$-models over $X\uplus\{x\}$.
- $\mathsf{P}_{\exists}^{s}$: To give a strategy on $\exists x.P,\Gamma(L,v)$ we must choose a value a for x and give a strategy on $P[a/x],\Gamma(L,v)$. By setting $a=v(s)$, we may use the interpretation of

the premise at (L, v).

- To interpret $\mathsf{P}_{\neq}$, we use the empty family of strategies, since there are no Θ-satisfying $\mathcal{L}$-models if Θ contains $x \neq x$.
- To interpret $\mathsf{P}_{\mathsf{ma}}^{x,y,z}$, we note that the collection of Θ-satisfying $\mathcal{L}$-models can be decomposed into those where x and y are identified (the left-hand premise) and those where they are distinct (the right-hand premise).
- To interpret $\mathsf{P}_{\mathsf{fneq}}$ we use the fact that all models satisfying $f(s) \neq f(t)$ also satisfy $s \neq t$.

We will present the formal semantics in due course. Strategies will be interpreted as *uniform* families, which must (in some sense) behave in the same manner across all components.

4.1.2 Embedding of FOILL

We can embed first-order Intuitionistic Linear Logic in WS1. This logic is defined by extending ILL with atoms $\phi(s)$ and quantifiers together with the rules given in Figure 4-2. Our logic WS1 allows the empty domain asa model and so we must consider a formulation of first order intuitionistic logic that admits the empty domain (so-called free logic, considered in [33]) using explicit variable sets.

Figure 4-2: Proof rules for FOILL — extends Figure 3-2

$$\frac{X \uplus \{x\}; \Gamma \vdash M}{X; \Gamma \vdash \forall x.M}\, x \notin FV(\Gamma) \qquad \frac{X; \Gamma, M[s/x] \vdash N}{X; \Gamma, \forall x.M \vdash N}\, FV(s) \subseteq X$$

Proposition 4.1.1 *For any proof p of $X; M_1, \ldots, M_n \vdash N$ in FOILL there is a proof $\kappa(p)$ in WS! of $X; \vdash N, M_1^{\perp}, \ldots, M_n^{\perp}$.*

Proof The right-$\forall$ rule corresponds to $\mathsf{P}_{\forall}$ and the left-$\forall$ rule corresponds to $\mathsf{P}_{\exists}^{\mathsf{T}}$. ∎

Equality

We demonstrate how the rules for equality can be used to derive reflexivity, symmetry and transitivity.

Reflexivity $x = x$:

$$\dfrac{\dfrac{}{x \neq x \vdash \bot}\,\mathsf{P}_{\neq}}{\vdash x = x}\,\mathsf{P}_{\mathsf{at}-}$$

Symmetry $y = x \multimap x = y$:

$$\dfrac{\dfrac{\dfrac{}{w \neq w \vdash \bot, w \neq w}\,\mathsf{P}_{\neq}}{\vdash w = w, w \neq w} \qquad \dfrac{\dfrac{\dfrac{}{y \neq x, x \neq y \vdash \top}}{y \neq x, x \neq y \vdash y \neq x}}{y \neq x, x \neq y \vdash \bot, y \neq x}}{\vdash x = y, y \neq x}\,\mathsf{P}_{\mathsf{ma}}^{y,x,w}$$

Transitivity $x = y \otimes y = z \multimap x = z$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x \neq w \vdash \top}}{x \neq w \vdash \top, w \neq w}}{x \neq w \vdash x \neq w, w \neq w}}{x \neq w \vdash x \neq w \parr w \neq w} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x \neq y, y \neq z \vdash \top}}{x \neq y, y \neq z \vdash \top, x \neq y}}{x \neq y, y \neq z \vdash y \neq z, x \neq y}}{x \neq y, y \neq z \vdash x \neq y \parr y \neq z}}{\dfrac{\dfrac{\dfrac{\dfrac{x \neq z \vdash x \neq y \parr y \neq z}{x \neq z \vdash \bot, x \neq y \parr y \neq z}}{x \neq z \vdash \bot, x \neq y, y \neq z}}{\vdash x = z, x \neq y, y \neq z}}\,\mathsf{P}_{\mathsf{ma}}^{y,z,w}$$

4.1.3 New Provable Formulas

In Section 2.2.3 we saw that the embedding of Intuitionistic Linear Logic into WS is not full: there are proofs that are not in the image of this translation. We can now strengthen this, by giving formulas that are not provable in Intuitionistic Linear Logic but are provable in $\mathsf{WS1}$.

Memoization

The formula $\phi_{\mathsf{ex}} = (\phi \& (\phi \multimap \bot)) \multimap \bot$ corresponds to an "additive excluded middle" in (negative) ILL, and is not provable. This formula is not provable in $\mathsf{WS1}$ either. However, consider the formula $\phi_{\mathsf{ex}} \multimap \phi_{\mathsf{ex}} \otimes \phi_{\mathsf{ex}}$. This is not provable in ILL but it *is* provable in $\mathsf{WS1}$ (as $\phi_{\mathsf{ex}} \otimes \phi_{\mathsf{ex}} \lhd \phi_{\mathsf{ex}}^{\perp}$). While Player can only access the input ϕ_{ex} once in the corresponding game, he can 'remember' whether ϕ was true or false, to give a winning history-sensitive (uniform) strategy. This proof is given by $\mathsf{P}_{\otimes}(p, p)$ where p is as follows:

$$\overline{\phi} \vdash \top \qquad\qquad \overline{\phi} \vdash \top$$

$$\overline{\phi} \vdash \overline{\phi} \qquad\qquad \overline{\phi} \vdash \overline{\phi}$$

$$\overline{\phi} \vdash \bot, \overline{\phi} \qquad\qquad \overline{\phi} \vdash \bot, \overline{\phi}$$

$$\overline{\phi} \vdash \bot, \phi, \overline{\phi} \qquad\qquad \overline{\phi} \vdash \bot, \phi, \overline{\phi}$$

$$\vdash \phi, \phi, \overline{\phi} \qquad\qquad \vdash \phi, \phi, \overline{\phi}$$

$$\vdash \phi \otimes \phi, \overline{\phi}$$

$$\overline{\phi} \vdash \top \qquad\qquad \vdash \top, \phi, \phi, \overline{\phi}$$

$$\overline{\phi} \vdash \overline{\phi} \qquad\qquad \vdash \top \oslash \phi, \phi, \overline{\phi}$$

$$\overline{\phi} \vdash \bot, \overline{\phi} \qquad\qquad \vdash \overline{\phi} \oplus (\top \oslash \phi), \phi, \overline{\phi}$$

$$\overline{\phi} \vdash \bot, \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \overline{\phi} \qquad\qquad \vdash \bot, (\overline{\phi} \oplus (\top \oslash \phi)), \phi, \overline{\phi}$$

$$\vdash \phi, \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \overline{\phi} \qquad\qquad \vdash \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \phi, \overline{\phi}$$

$$\vdash \phi \otimes (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \overline{\phi}$$

$$\vdash (\phi \otimes (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi))) \lhd \overline{\phi}$$

$$\vdash \top, \phi, (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \overline{\phi}$$

$$\vdash \top \oslash \phi, (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \overline{\phi}$$

$$\vdash \overline{\phi} \oplus (\top \oslash \phi), (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \overline{\phi}$$

$$\vdash \overline{\phi} \parr ((\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)))$$

$$\vdash \bot, \overline{\phi}, (\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)))$$

$$p_1 \qquad\qquad \vdash \bot \lhd \overline{\phi}, (\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)))$$

$$\vdash (\phi \& (\bot \lhd \overline{\phi})), (\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)))$$

$$\vdash (\phi \& (\bot \lhd \overline{\phi})) \lhd ((\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi))))$$

$$\vdash \top, (\phi \& (\bot \lhd \overline{\phi})), (\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)))$$

$$\vdash \top \oslash (\phi \& (\bot \lhd \overline{\phi})), (\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)))$$

$$\vdash (\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi))) \parr (\top \oslash (\phi \& (\bot \lhd \overline{\phi})))$$

$$\vdash \bot, (\overline{\phi} \oplus (\top \oslash \phi)), \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \top \oslash (\phi \& (\bot \lhd \overline{\phi}))$$

$$\vdash \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)), \top \oslash (\phi \& (\bot \lhd \overline{\phi}))$$

where p_1 is:

$$\overline{\phi} \vdash \top$$

$$\overline{\phi} \vdash \overline{\phi}$$

$$\overline{\phi} \vdash \overline{\phi} \oplus (\top \oslash \phi)$$

$$\overline{\phi} \vdash \bot, \overline{\phi} \oplus (\top \oslash \phi)$$

$$\overline{\phi} \vdash \top, \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi))$$

$$\overline{\phi} \vdash \overline{\phi}, \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi))$$

$$\overline{\phi} \vdash \overline{\phi} \oplus (\top \oslash \phi), \bot \lhd (\overline{\phi} \oplus (\top \oslash \phi))$$

$$\overline{\phi} \vdash \bot, (\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)))$$

$$\vdash \phi, (\overline{\phi} \oplus (\top \oslash \phi)) \oslash (\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi)))$$

This example can be extended to the exponentials: while $\phi_{\mathsf{ex}} \multimap !\phi_{\mathsf{ex}}$ is not provable in ILL, its translation is provable in WS1 (it is simply $\mathsf{P}_{\mathsf{ana}}(p)$).

Remark Note that $\phi \multimap \phi \otimes \phi$ is also provable in WS1 but not ILL — but this is for less interesting reasons, as it only makes use of the fact that atoms are interpreted as one-move games and local alternation.

Medial Rule

In Section 2.2.3 we described a family of formulas

$$((A \otimes B \multimap \bot) \otimes (C \otimes D \multimap \bot) \multimap \bot) \multimap$$
$$((\mathrm{A} \multimap \bot) \otimes (C \multimap \bot) \multimap \bot) \otimes ((B \multimap \bot) \otimes (D \multimap \bot) \multimap \bot)$$

that are not provable in ILL but are provable in WS1. We can now use negative atoms to formalise this.

Proposition 4.1.2 *Let $\alpha, \beta, \gamma, \delta$ be negative (nullary) atoms. Then $\vdash ((\alpha \otimes \beta \multimap \bot) \otimes (\gamma \otimes \delta \multimap \bot) \multimap \bot) \multimap ((\alpha \multimap \bot) \otimes (\gamma \multimap \bot) \multimap \bot) \otimes ((\beta \multimap \bot) \otimes (\delta \multimap \bot) \multimap \bot)$ is not provable in ILL but it is provable in WS1.*

As noted in [14], this formula is not provable in ILL. We can use the cut-elimination theorem of MLL to perform an exhaustive proof search on this sequent and find that it does not lead to a proof. Alternatively, we see in [6] that multiplicative sequents of Intuitionistic Linear Logic can be represented as uniform history-free strategies on the underlying game, and an exhaustive search shows that there are no such strategies on this game.

However, there is an evident uniform history-sensitive strategy. For example, if Opponent first chooses the left hand component in the output and the right hand component in the input, Player can choose to play copycat between the copies of γ, and so on. There is a proof in WS1 denoting this strategy. A branch of the proof is given below. The use of the $\mathsf{P}_{\otimes}$ demonstrates where the proof branches; there are four branches corresponding to the two uses of $\mathsf{P}_{\otimes}$.

$$\overline{\alpha},\overline{\gamma} \vdash \top$$

$$\overline{\alpha},\overline{\gamma} \vdash \top,\overline{\delta}$$

$$\overline{\alpha},\overline{\gamma} \vdash \overline{\gamma},\overline{\delta}$$

$$\overline{\alpha},\overline{\gamma} \vdash \overline{\gamma}\parr\overline{\delta}$$

$$\overline{\alpha},\overline{\gamma} \vdash \bot,(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\gamma}\parr\overline{\delta}$$

$$\overline{\alpha} \vdash \gamma,(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\gamma}\parr\overline{\delta}$$

$$\overline{\alpha} \vdash \top,\gamma,(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\gamma}\parr\overline{\delta}$$

$$\overline{\alpha} \vdash (\top\oslash\gamma),(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\gamma}\parr\overline{\delta}$$

$$\overline{\alpha} \vdash (\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\gamma}\parr\overline{\delta}$$

$$\overline{\alpha} \vdash \bot,\overline{\gamma}\parr\overline{\delta},(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\overline{\alpha} \vdash (\bot\lhd\overline{\gamma}\parr\overline{\delta}),(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\overline{\alpha} \vdash \top,\overline{\beta},(\bot\lhd\overline{\gamma}\parr\overline{\delta}),(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\overline{\alpha} \vdash \overline{\alpha},\overline{\beta},(\bot\lhd\overline{\gamma}\parr\overline{\delta}),(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\mathsf{P}_{\parr 1}\quad \overline{\alpha} \vdash \overline{\alpha}\parr\overline{\beta},(\bot\lhd\overline{\gamma}\parr\overline{\delta}),(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\overline{\alpha} \vdash \overline{\alpha}\parr\overline{\beta}\oslash(\bot\lhd\overline{\gamma}\parr\overline{\delta}),(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\overline{\alpha} \vdash \bot,\top\oslash\gamma,(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\alpha}\parr\overline{\beta}\oslash(\bot\lhd\overline{\gamma}\parr\overline{\delta})$$

$$\vdash \alpha,\top\oslash\gamma,(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\alpha}\parr\overline{\beta}\oslash(\bot\lhd\overline{\gamma}\parr\overline{\delta})$$

$$\vdash \top,\alpha,\top\oslash\gamma,(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\alpha}\parr\overline{\beta}\oslash(\bot\lhd\overline{\gamma}\parr\overline{\delta})$$

$$\vdash \top\oslash\alpha,\top\oslash\gamma,(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\alpha}\parr\overline{\beta}\oslash(\bot\lhd\overline{\gamma}\parr\overline{\delta})$$

$$\mathsf{P}_{\parr 1}\quad \vdash (\top\oslash\alpha)\parr(\top\oslash\gamma),(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\alpha}\parr\overline{\beta}\oslash(\bot\lhd\overline{\gamma}\parr\overline{\delta})$$

$$\vdash (\top\oslash\alpha)\parr(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\overline{\alpha}\parr\overline{\beta}\oslash(\bot\lhd\overline{\gamma}\parr\overline{\delta})$$

$$\vdash \bot,\overline{\alpha}\parr\overline{\beta},(\bot\lhd\overline{\gamma}\parr\overline{\delta}),(\top\oslash\alpha)\parr(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\vdash \bot\lhd\overline{\alpha}\parr\overline{\beta},(\bot\lhd\overline{\gamma}\parr\overline{\delta}),(\top\oslash\alpha)\parr(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)) \qquad \vdots$$

$$\mathsf{P}_{\otimes}\quad \vdash (\bot\lhd\overline{\alpha}\parr\overline{\beta})\otimes(\bot\lhd\overline{\gamma}\parr\overline{\delta}),(\top\oslash\alpha)\parr(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\vdash \top,((\bot\lhd\overline{\alpha}\parr\overline{\beta})\otimes(\bot\lhd\overline{\gamma}\parr\overline{\delta})),(\top\oslash\alpha)\parr(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\vdash \top\oslash((\bot\lhd\overline{\alpha}\parr\overline{\beta})\otimes(\bot\lhd\overline{\gamma}\parr\overline{\delta})),(\top\oslash\alpha)\parr(\top\oslash\gamma)\oslash(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta))$$

$$\vdash \bot,(\top\oslash\alpha)\parr(\top\oslash\gamma),(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\top\oslash((\bot\lhd\overline{\alpha}\parr\overline{\beta})\otimes(\bot\lhd\overline{\gamma}\parr\overline{\delta}))$$

$$\vdash (\bot\lhd(\top\oslash\alpha)\parr(\top\oslash\gamma)),(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\top\oslash((\bot\lhd\overline{\alpha}\parr\overline{\beta})\otimes(\bot\lhd\overline{\gamma}\parr\overline{\delta})) \qquad \vdots$$

$$\mathsf{P}_{\otimes}\quad \vdash (\bot\lhd(\top\oslash\alpha)\parr(\top\oslash\gamma))\otimes(\bot\lhd(\top\oslash\beta)\parr(\top\oslash\delta)),\top\oslash((\bot\lhd\overline{\alpha}\parr\overline{\beta})\otimes(\bot\lhd\overline{\gamma}\parr\overline{\delta}))$$

Here is another formula that is provable in WS1 but not in ILL:

$$[\alpha\otimes(\gamma\&\delta)]\&[\beta\otimes(\gamma\&\delta)]\&[(\alpha\&\beta)\otimes\gamma]\&[(\alpha\&\beta)\otimes\delta]\multimap(\alpha\&\beta)\otimes(\gamma\&\delta)$$

The derivation in WS was given in Section 2.2.3, and can be specialised to axioms.

4.1.4 Imperative Objects

We have already represented imperative objects in our logic; we next show how the first-order structure enriches these ideas.

Data-Independent Programming

In Section 3.4.3 we used the exponential in WS1 to represent Boolean cells and stacks. We can use our quantifiers to represent *data-independent* cells and stacks, where the underlying ground type at a given $\mathcal{L}$-structure L is $|L|$. As a formula/game, this is represented by $\mathbf{V} = \bot \lhd \exists x.\top$ (a dialogue in this game consists of Opponent playing a question move q and Player responding with an element of $|L|$).

Let $\mathbf{Vi} = \forall x.\bot \lhd \top$. We next give the proof for a data-independent stack, parametrised by a starting stack $!\mathbf{V} \multimap !(\mathbf{V}\&\mathbf{Vi})$.

$$\dfrac{\dfrac{\dfrac{\mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\mathsf{P}^{!}_{\mathsf{con}}\ \mathsf{P}^{!}_{\mathsf{der}}\dfrac{\vdash !(\bot \lhd \exists x.\top), !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}{\vdash \bot \lhd \exists x.\top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P}_{\mathsf{id}}\dfrac{}{\{x\};\vdash !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\{x\};\vdash \top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}\mathsf{P}^{x}_{\exists}}{\{x\};\vdash \exists x.\top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\{x\};\vdash \bot, \exists x.\top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\{x\};\vdash \bot \lhd \exists x.\top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\{x\};\vdash !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\{x\};\vdash \top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\{x\};\vdash \bot, \top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\vdash \forall x.\bot, \top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\vdash \forall x.\bot \lhd \top, !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}}{\mathsf{P}_{\mathsf{ana}}\dfrac{\vdash (\bot \lhd \exists x.\top)\&(\forall x.\bot \lhd \top), !(\bot \lhd \exists x.\top), ?(\top \oslash \forall x.\bot)}{\vdash !((\bot \lhd \exists x.\top)\&(\forall x.\bot \lhd \top)), ?(\top \oslash \forall x.\bot)}}$$

Good Variable

In Section 3.4.3 we represented a Boolean reference cell in WS1 on the formula $!(\mathbf{B}\&\mathbf{Bi})$. However, there are other proofs of this formula that do not behave like a standard reference cell: for example, the `read` method may always return `tt` regardless of what was written. This is a *bad variable* [8]. We can use uniformity of the semantics to define formulas for which all proofs denote good variables, albeit variables that can only be written to once.

The formula $\mathbf{worm} = \mathbf{Bi} \oslash !\mathbf{B}$ represents a Boolean variable which can be written once, then read many times. One proof/strategy of this formula will be a valid Boolean cell: if Opponent plays `inputX` then Player responds with `ok`, if Opponent then tries to read the cell `q`, then Player responds with `X`. But there are also bad variables.

To exclude such behaviour, we can replace the input/output moves with atoms. Define $\mathbf{B}' = \bot \lhd (\overline{\phi} \oplus \overline{\psi})$ and $\mathbf{Bi}' = (\phi \& \psi) \lhd \top$, with $\mathbf{worm}' = \mathbf{Bi}' \oslash !\mathbf{B}'$. If ϕ and ψ are assigned `ff`, then this denotes the same dialogue as **worm**. However, the denotation of any proof of $\mathbf{worm}'$ at such a model must be the good variable strategy. The rule for atoms (and semantically, uniformity of strategies) ensures that moves in ϕ must be played before $\overline{\phi}$, and ψ before $\overline{\psi}$. Resultantly, Player can only respond with a particular Boolean in the `read` component if it has previously been given as an input in the `write` component. A proof of this formula is given below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\phi} \vdash \top}}{\overline{\phi} \vdash \overline{\phi}}\,\mathsf{P}_{\oplus 1}}{\overline{\phi} \vdash \overline{\phi} \oplus \overline{\psi}}}{\overline{\phi} \vdash \bot, \overline{\phi} \oplus \overline{\psi}}}{\overline{\phi} \vdash \bot \lhd \overline{\phi} \oplus \overline{\psi}}\,\mathsf{P}_{\mathsf{prom}}}{\overline{\phi} \vdash\, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}}{\overline{\phi} \vdash \top, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}}{\overline{\phi} \vdash \bot, \top, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}}{\vdash \phi, \top, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\psi} \vdash \top}}{\overline{\psi} \vdash \overline{\psi}}\,\mathsf{P}_{\oplus 2}}{\overline{\psi} \vdash \overline{\phi} \oplus \overline{\psi}}}{\overline{\psi} \vdash \bot, \overline{\phi} \oplus \overline{\psi}}}{\overline{\psi} \vdash \bot \lhd \overline{\phi} \oplus \overline{\psi}}\,\mathsf{P}_{\mathsf{prom}}}{\overline{\psi} \vdash\, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}}{\overline{\psi} \vdash \top, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}}{\overline{\psi} \vdash \bot, \top, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}}{\vdash \psi, \top, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}$$

$$\dfrac{\vdash (\phi \& \psi), \top, !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}{\vdash ((\phi \& \psi) \lhd \top) \oslash !(\bot \lhd \overline{\phi} \oplus \overline{\psi})}$$

We can consider a further example: an object with two methods, a `switch` procedure and a `read` method that returns a Boolean, denoted by the formula $!((\bot \lhd \top) \& \mathbf{B})$. We can refine this formula to only allow strategies satisfying the following property: if the `read` method returns `true`, then the switch has previously been invoked. The appropriate formula is $(\phi \lhd \top) \otimes (\bot \lhd (\overline{\phi} \oplus \top))$. We cannot describe a property that requires that the read method returns `true` just when the switch has been invoked, since the only kind of specifications we can express in this way are of the form 'if Player plays move X, Opponent has previously played move Y'. It is for this reason that the good variable example does not scale to Boolean cells that admit multiple `write` operations.

Refinements and Specifications

The formulas available in WS1 allow us to specify the behaviour of a program in more detail than its programming language type. For example, functions $\mathbf{V} \multimap \mathbf{V}$ in context $X; \Theta$ are represented as proofs of $X; \Theta \vdash \mathbf{V}, \mathbf{V}^{\perp}$. We can consider formulas that represent a subgame of the semantics of this sequent. For example, we can consider $\mathbf{Id} = \bot \lhd (\top \oslash \forall y.(\bot \lhd \exists x.y = x))$ and a total strategy (or proof) representing the

embedding $\mathbf{Id} \multimap (\mathbf{V} \multimap \mathbf{V})$. The formula $\mathbf{Id}$ represents a type of identity maps on $\mathbf{V}$ — it is in some sense a form of first-order dependent type. We can view the formula $\mathbf{Id}$ as a *specification* on $\mathbf{V} \multimap \mathbf{V}$ — a program satisfies it if it factors through the embedding. In Chapter 5 we will explore these ideas further.

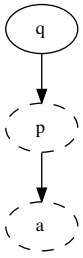
4.2 Semantics of WS1

We next give formal semantics to proofs in WS1.

4.2.1 Uniform Strategies

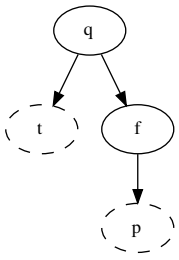
We have seen that a sequent $X; \Theta \vdash \Gamma$ of WS1 can be interpreted as a family of games, indexed over Θ-satisfying $\mathcal{L}$-models over X. We interpret a proof of $X; \Theta \vdash \Gamma$ as a family of strategies on the appropriate family of games. However, the strategies that are the interpretation of a proof are *uniform* in behaviour.

For example, the family denoted by $\top \oslash (\phi \lhd \top)$ has games of the following form:



Here we represent the forest of plays directly. The moves in dotted circles are only available if $(L, v) \models \overline{\phi}$. There is a unique total strategy on the (positive) game above in both cases, and this family is uniform in the sense that the strategy on models which satisfy ϕ is a *substrategy* of the strategy on models satisfying $\overline{\phi}$ — if $(L, v) \models \phi$ and $(L', v') \models \overline{\phi}$ then $\sigma_{[\![\top \oslash (\phi \lhd \top)]\!](L,v)} \subseteq \sigma_{[\![\top \oslash (\phi \lhd \top)]\!](L',v')}$.

In contrast, consider the formula $\bot \lhd (\overline{\phi} \oplus (\top \oslash \phi))$. The game forest is given as follows, using the same notation as above:



There is a family of strategies on this (negative) game: if ϕ is true, Player plays f and if $\overline{\phi}$ is true, Player plays $\mathtt{t}$. However, this strategy is not uniform as the choice of second move depends on the truth value of ϕ in the appropriate $\mathcal{L}$-structure. Dually, the formula is not provable in $\mathsf{WS1}$.

In this section we formalise uniformity of strategies.

Game Embeddings

To formalise subgames and uniformity categorically, we use the following machinery of embedding-projection pairs:

Definition Let $\mathcal{C}$ be a poset-enriched category. The category $\mathcal{C}_e$ has the same objects as $\mathcal{C}$ and a map $A \to B$ in $\mathcal{C}_e$ consists of a pair (i_f, p_f) where $i_f : A \to B$ and $p_f : B \to A$ in $\mathcal{C}$, such that $p_f \circ i_f = \mathsf{id}$ and $i_f \circ p_f \sqsubseteq \mathsf{id}$.

- The identity is given by $(\mathsf{id}, \mathsf{id})$.
- For composition, set $(i_f, p_f) \circ (i_g, p_g) = (i_f \circ i_g, p_g \circ p_f)$. We need to check this is a valid pairing: $p_{f \circ g} \circ i_{f \circ g} = p_g \circ p_f \circ i_f \circ i_g = p_g \circ \mathsf{id} \circ i_g = \mathsf{id}$ and $i_{f \circ g} \circ p_{f \circ g} = i_f \circ i_g \circ p_g \circ p_f \sqsubseteq i_f \circ \mathsf{id} \circ p_f = i_f \circ p_f \sqsubseteq \mathsf{id}$.
- It is clear that composition is associative and that $f = f \circ \mathsf{id} = \mathsf{id} \circ f$.

Let $\mathcal{G}$ denote the poset-enriched category of win-games and (not-necessarily winning) strategies, and $\mathcal{G}_s$ its subcategory of strict strategies, with $\sqsubseteq$ given by strategy inclusion. A forest embedding of A into B corresponds to a map $A \to B$ in $\mathcal{G}_e$.

Remark Note that morphisms in $\mathcal{G}_e$ are not winning. The reason for this is that strategies that realise forest embeddings need not be total (e.g. the embedding $I \to \bot$). A consequence of this is that the embeddings ignore winning conditions. To consider embeddings of the winning condition also, we could have required that maps $\sigma \in \mathcal{G}_e(A, B)$ satisfy the second condition of the definition of winningness in Section 3.2.3. However, we will not need this condition here.

Proposition 4.2.1 *If $f : A \to B$ in $\mathcal{G}_e$ then i_f and p_f are strict.*

Proof If i_f responds to an opening move in B with a move in B then so does $i_f \circ p_f$ and so $i_f \circ p_f \sqsubseteq \mathsf{id}$ fails. Similarly, if p_f responds to an opening move in A with a move in A then so does $p_f \circ i_f$ and so $p_f \circ i_f = \mathsf{id}$ fails. ∎

We can thus define identity-on-objects functors $i : \mathcal{G}_e \to \mathcal{G}_s$ and $p : \mathcal{G}_e \to \mathcal{G}_s^{\mathsf{op}}$ each selecting the appropriate component of the embedding.

In fact, we know something stronger about i_f and p_f: the components of any embedding are of *zig-zag* shape [54] i.e. the strategy responds to *every* move in A (resp. B) with a move in B (resp. A) if it responds at all.

We can show that our operations on win-games lift to functors on $\mathcal{G}_e$.

Proposition 4.2.2 *Each of the operations* $\multimap, \oslash, \otimes, \&, !$ *extend to covariant (bi)functors on* $\mathcal{G}_e$.

Proof

- We set $(i,p)\otimes(i',p') = (i\otimes i', p\otimes p')$. Then $(p\otimes p')\circ(i\otimes i') = (p\circ i)\otimes(p'\circ i') = \mathsf{id}\otimes\mathsf{id} = \mathsf{id}$ and $(i\otimes i')\circ(p\otimes p') = (i\circ p)\otimes(i'\circ p') \sqsubseteq \mathsf{id}\otimes\mathsf{id} = \mathsf{id}$ as required. The case for $\&$ and $\oslash$ are similar, as they form monotonic bifunctors on $\mathcal{G}_s$.
- We set $(i,p)\multimap(i',p') = (p\multimap i', i\multimap p')$. Then $(i\multimap p')\circ(p\multimap i') = (p\circ i)\multimap(p'\circ i') = \mathsf{id}\multimap\mathsf{id} = \mathsf{id}$ and $(p\multimap i')\circ(i\multimap p') = (i\circ p)\multimap(i'\circ p') \sqsubseteq \mathsf{id}\multimap\mathsf{id} = \mathsf{id}$.
- We set $!(i,p) = (!i, !p)$. Then $!p\circ!i = !(p\circ i) = !\mathsf{id} = \mathsf{id}$ and $!i\circ!p = !(i\circ p) \sqsubseteq !\mathsf{id} = \mathsf{id}$. ∎

Lax natural Transformations

Given an embedding $e : A \to B$ and strategies $\sigma_A : A$, $\sigma_B : B$, σ_B *restricts to* σ_A if $\sigma_A = p_e \circ \sigma_B$. We generalise this idea using lax natural transformations.

Definition Let $\mathcal{C}$ be a category, $\mathcal{D}$ a poset-enriched category and $F, G : \mathcal{C} \to \mathcal{D}$. A *lax natural transformation* $F \Rightarrow G$ is a family of arrows $\mu_A : F(A) \to G(A)$ such that $\eta_B \circ F(f) \sqsupseteq G(f) \circ \eta_A$.

$$\begin{array}{ccc} F(A) & \xrightarrow{\mu_A} & G(A) \\ {\scriptstyle F(f)}\downarrow & \sqsupseteq & \downarrow{\scriptstyle G(f)} \\ F(B) & \xrightarrow[\mu_B]{} & G(B) \end{array}$$

It is clear that natural transformations are lax natural, by reflexivity of $\sqsubseteq$. We can compose lax natural transformations using vertical composition:

$$\begin{array}{ccccc} F(A) & \xrightarrow{\mu_A} & G(A) & \xrightarrow{\eta_A} & H(A) \\ {\scriptstyle F(f)}\downarrow & \sqsupseteq & \downarrow{\scriptstyle G(f)} & \sqsupseteq & \downarrow{\scriptstyle H(f)} \\ F(B) & \xrightarrow[\mu_B]{} & G(B) & \xrightarrow[\eta_B]{} & H(B) \end{array}$$

There is also a form of horizontal composition, provided that one of the two functors is the identity.

Proposition 4.2.3 *Let $H, G : \mathcal{C} \to \mathcal{D}$ and $\mu : G \Rightarrow H$ a lax natural transformation.*

- *If $F : \mathcal{B} \to \mathcal{C}$ then there is a lax natural transformation $\mu F : G \circ F \Rightarrow H \circ F$.*
- *If $J : \mathcal{D} \to \mathcal{E}$ is monotonic then there is a lax natural transformation $J\mu : J \circ G \to J \circ H$.*

Proof Define $(\mu F)_A = \mu_{F(A)}$. Then $(\mu F)_B \circ GF(f) = \mu_{F(B)} \circ GF(f) \sqsupseteq HF(f) \circ \mu_{F(A)} = HF(f) \circ (\mu F)_A$.

Similarly, let $(J\mu)_A = J(\mu_A)$. Then $(J\mu)_B \circ JG(f) = J(\mu_B \circ G(f)) \sqsupseteq J(H(f) \circ \mu_A) = JH(f) \circ J(\mu_A) = JH(f) \circ (J\mu)_A$ as required. ∎

Uniform Winning Strategies

Definition Let $F, G : \mathcal{C} \to \mathcal{G}_e$. A *uniform strategy* from F to G is a lax natural transformation $\sigma : i \circ F \Rightarrow i \circ G$. A *uniform total strategy* is a uniform strategy σ where each σ_A is total. A *uniform winning strategy* is a uniform strategy where each σ_A is winning.

If $f : A \to B$, the lax naturality condition requires $i_{G(f)} \circ \sigma_A \sqsubseteq \sigma_B \circ i_{F(f)}$. Thus $\sigma_A = p_{G(f)} \circ i_{G(f)} \circ \sigma_A \sqsubseteq p_{G(f)} \circ \sigma_B \circ i_{F(f)}$. But since σ_A is total, it is maximal in the ordering $\sqsubseteq$ and we must have $\sigma_A = p_{G(f)} \circ \sigma_B \circ i_{F(f)}$. Similarly, we see that $\sigma_A = p_{G(f)} \circ \sigma_B \circ i_{F(f)}$ implies the lax naturality condition as $i_{G(f)} \circ \sigma_A = i_{G(f)} \circ p_{G(f)} \circ \sigma_B \circ i_{F(f)} \sqsubseteq \sigma_B \circ i_{F(f)}$. Thus, lax naturality captures the fact that σ_A is determined by σ_B via restriction. If F is the constant functor κ_I, this reduces to $\sigma_A = p_{G(f)} \circ \sigma_B$.

We can construct a WS-category of uniform strategies over a base category $\mathcal{C}$. Let $\mathcal{G}^{\mathcal{C}}$ be the category where:

- Objects are functors $\mathcal{C} \to \mathcal{G}_e$
- An arrow $F \to G$ is a uniform strategy $F \Rightarrow G$
- Composition is given by vertical composition of lax natural transformations
- The identity on a functor F is given by the lax natural transformation $\eta : F \Rightarrow F$ where $\eta_A = \mathsf{id}_{F(A)}$.

Similarly, we can construct a category $\mathcal{W}^{\mathcal{C}}$ of functors and uniform winning strategies.

Proposition 4.2.4 *$\mathcal{G}^{\mathcal{C}}$ is a WS!-category.*

Proof • **Symmetric monoidal category:** $F \otimes G$ is defined to be $\otimes \circ (F \times G) \circ \Delta$ where $\Delta : \mathcal{C} \to \mathcal{C} \times \mathcal{C}$ is the diagonal. So, $(F \otimes G)(A) = F(A) \otimes G(A)$. On arrows, we set $(\eta \otimes \rho)_A = \eta_A \otimes \rho_A$. We need to show that if $f : L \to K$ then $(i_{A(f)} \otimes i_{C(f)}) \circ (\eta_K \otimes \rho_K) \sqsupseteq (i_{B(f)} \otimes i_{D(f)}) \circ (\eta_L \otimes \rho_L)$. That is, we need to show that $(i_{A(f)} \circ \eta_K) \otimes (i_{C(f)} \circ \rho_K) \sqsupseteq (i_{B(f)} \circ \eta_L) \otimes (i_{D(f)} \circ \rho_L)$. But this is clear by lax naturality of η and ρ and monotonicity of $\otimes$.

The tensor unit I is the constant functor, sending all objects to the game I and arrows to id_I.

The morphisms assoc, $\mathsf{runit}_\otimes$, $\mathsf{lunit}_\otimes$ and sym are defined pointwise: e.g. $(\mathsf{assoc}_{F,G,H})_X = \mathsf{assoc}_{F(X),G(X),H(X)}$. To check for lax naturality, we must use horizontal composition. For example, consider the map $\mathsf{assoc} : (F \otimes G) \otimes H \to F \otimes (G \otimes H)$ defined pointwise as described. The domain is $(F \otimes G) \otimes H = ((\_ \otimes \_) \otimes \_) \circ (i \circ F \times i \circ G \times i \circ H) \circ \Delta_3$ where Δ_3 denotes the diagonal functor $\mathcal{C} \to \mathcal{C} \times \mathcal{C} \times \mathcal{C}$. Similarly, the codomain is $(\_ \otimes (\_ \otimes \_)) \circ (i \circ F \times i \circ G \times i \circ H) \circ \Delta_3$. We can thus see that assoc is equal to the horizontal composition $\mathsf{assoc}J$ where $J = (i \circ F \times i \circ G \times i \circ H) \circ \Delta_3$ and assoc is the natural transformation $\_ \otimes (\_ \otimes \_) \Rightarrow (\_ \otimes \_) \otimes \_$ in $\mathcal{G}_s$.

$$\begin{array}{ccccccc}
\mathcal{C} & \xrightarrow{\Delta_3} & \mathcal{C} \times \mathcal{C} \times \mathcal{C} & \xrightarrow{i \circ F \times i \circ G \times i \circ H} & \mathcal{G}_s \times \mathcal{G}_s \times \mathcal{G}_s & \xrightarrow{_ \otimes (_ \otimes _)} & \mathcal{G}_s \\
 & \mathsf{id} \Downarrow & & \mathsf{id} \Downarrow & & \mathsf{assoc} \Downarrow & \\
\mathcal{C} & \xrightarrow{\Delta_3} & \mathcal{C} \times \mathcal{C} \times \mathcal{C} & \xrightarrow{i \circ F \times i \circ G \times i \circ H} & \mathcal{G}_s \times \mathcal{G}_s \times \mathcal{G}_s & \xrightarrow{(_ \otimes _) \otimes _} & \mathcal{G}_s
\end{array}$$

One can similarly express the other monoidal isomorphisms in this way to see lax naturality. The coherence equations lift pointwise from $\mathcal{G}$.

- **Symmetric monoidal closed category:** We have seen that $\multimap$ extends to a covariant bifunctor on $\mathcal{G}_e$. The object $F \multimap G$ is defined to be $\multimap \circ (F \times G) \circ \Delta$. We define $\Lambda : \mathcal{G}^{\mathcal{C}}(F \otimes G, H) \to \mathcal{G}^{\mathcal{C}}(F, G \multimap H)$ pointwise by $\Lambda(\eta)_A = \Lambda(\eta_A)$.

  We need to show that if $\eta : F \otimes G \Rightarrow H$ is lax natural then so is $\Lambda(\eta)$. If $f : A \to B$ we need to show that $\Lambda(\eta_B) \circ i_{F(f)} \sqsupseteq (p_{G(f)} \multimap i_{H(f)}) \circ \Lambda(\eta_A)$. But $\Lambda(\eta_B) \circ i_{F(f)} = \Lambda(\eta_B \circ (i_{F(f)} \otimes \mathsf{id}))$ and $(p_{G(f)} \multimap i_{H(f)}) \circ \Lambda(\eta_A) = \Lambda(i_{H(f)} \circ \eta_A \circ (\mathsf{id} \otimes p_{G(f)}))$ and so it is sufficient to show that $\Lambda(\eta_B \circ (i_{F(f)} \otimes \mathsf{id})) \sqsupseteq \Lambda(i_{H(f)} \circ \eta_A \circ (\mathsf{id} \otimes p_{G(f)}))$. By monotonicity of Λ, it is sufficient to show that $\eta_B \circ (i_{F(f)} \otimes \mathsf{id}) \sqsupseteq i_{H(f)} \circ \eta_A \circ (\mathsf{id} \otimes p_{G(f)})$. But $\eta_B \circ (i_{F(f)} \otimes \mathsf{id}) \sqsupseteq \eta_B \circ (i_{F(f)} \otimes \mathsf{id}) \circ (\mathsf{id} \otimes i_{G(f)}) \circ (\mathsf{id} \otimes p_{G(f)}) \sqsupseteq i_{H(f)} \circ \eta_A \circ (\mathsf{id} \otimes p_{G(f)})$ using lax naturality of η, as required.

  The equation $\Lambda(F \circ G \circ (H \otimes J)) = (J \multimap F) \circ \Lambda(G) \circ H$ which shows that Λ is a natural transformation of hom sets lifts pointwise from $\mathcal{G}$. The fact that Λ defines

an isomorphism of hom sets also inherits from $\mathcal{G}$, defining $\Lambda^{-1}(\eta)_A = \Lambda^{-1}(\eta_A)$.

- **Inclusive sequoidal category:** The subcategory $\mathcal{G}_s^{\mathcal{C}}$ is defined to be the uniform strategies that are pointwise strict. Then $F \oslash G$ is defined to be $\oslash \circ (F \times G) \circ \Delta$ and $\eta \oslash \rho$ is defined pointwise. We can see that $\eta \oslash \rho$ is lax natural from monotonicity of $\oslash$ on arrows. The sequoidal natural (iso)morphisms inherit from $\mathcal{G}$, and once again we can show lax naturality using horizontal composition.
- **Products:** The product operation on objects of $\mathcal{G}^{\mathcal{C}}$ is defined to be $\times' \circ (F \times G) \circ \Delta$ where $\times'$ represents the product bifunctor on $\mathcal{G}_e$ (previously denoted $\times$). The projections are defined pointwise. To see that they are uniform strategies (in particular lax natural) we note that $\pi_i : F_1 \times F_2 \Rightarrow F_i$ can be defined using horizontal composition. If $\eta : F \Rightarrow G$ and $\rho : F \Rightarrow H$ then $\langle \eta, \rho \rangle : F \Rightarrow G \times H$ is defined by $\langle \eta, \rho \rangle_A = \langle \eta_A, \rho_A \rangle$ using pairing in $\mathcal{G}$. The fact that the universal property is satisfied lifts from $\mathcal{G}$. $\mathcal{G}_s^{\mathcal{C}}$ also has products, preserved by the inclusion functor into $\mathcal{G}^{\mathcal{C}}$.
- **Decomposable and Distributive:** The decomposability and distributivity axioms inherit from pointwise from $\mathcal{G}$.
- **Sequoidal closed category:** We need to show that the operation $\Lambda_s : f \mapsto \Lambda(f \circ \mathsf{wk}) : \mathcal{G}_s^{\mathcal{C}}(B \oslash A, C) \to \mathcal{G}_s^{\mathcal{C}}(B, A \multimap C)$ is an isomorphism. Given $\eta : B \oslash A \to C$ in $\mathcal{G}_s^{\mathcal{C}}$ we can construct $\Lambda_s^{-1}(\eta) : \mathcal{G}_s^{\mathcal{C}}(B, A \multimap C) \Rightarrow \mathcal{G}_s^{\mathcal{C}}(B \oslash A, C)$ by $\Lambda_s^{-1}(\eta)_A = \Lambda_s^{-1}(\eta_A)$ using sequoidal closure of $\mathcal{G}$.
- **Linear functional extensionality:** The fact that lfe has an inverse lifts pointwise from $\mathcal{G}$.
- **Coalgebraic exponential comonoid:** We need to show that the functor $F \oslash \_$ has a final coalgebra $!F$ for each F. The object $!F$ is simply $! \circ F$, and we set $(\alpha_F)_A = \alpha_{F(A)}$. We can show that α is lax natural using horizontal composition. Given $\eta : G \Rightarrow F \oslash G$ we define $(\!|\, \eta \,|\!) : G \Rightarrow !F$ to be $(\!|\, \eta \,|\!)_A = (\!|\, \eta_A \,|\!)$. We need to show that $(\!|\, \eta \,|\!)$ is lax natural.

  Let $A, B, C, D \in \mathcal{G}$, $f : B \to D$ and strict $g : A \to C$. We need to show that $(g \oslash f) \circ \sigma_{AB} \sqsubseteq \sigma_{CD} \circ f$ implies $!g \circ (\!|\, \sigma_{AB} \,|\!) \sqsubseteq (\!|\, \sigma_{CD} \,|\!) \circ f$.

  We first show that $!g \circ (\!|\, \sigma_{AB} \,|\!) = (\!|\, (g \oslash \mathsf{id}) \circ \sigma_{AB} \,|\!)$. We know that $!g = (\!|\, (g \oslash \mathsf{id}) \circ \alpha \,|\!)$ and so by fusion it is sufficient to show that $(g \oslash \mathsf{id}) \circ \alpha \circ (\!|\, \sigma_{AB} \,|\!) = (\mathsf{id} \oslash (\!|\, \sigma_{AB} \,|\!)) \circ (g \oslash \mathsf{id}) \circ \sigma_{AB}$. But we know that $\alpha \circ (\!|\, \sigma_{AB} \,|\!) = (\mathsf{id} \oslash (\!|\, \sigma_{AB} \,|\!)) \oslash \sigma_{AB}$ (cancellation law) so this is clear.

  We thus need to show that $(\!|\, (g \oslash \mathsf{id}) \circ \sigma_{AB} \,|\!) \sqsubseteq (\!|\, \sigma_{CD} \,|\!) \circ f$. By Proposition 3.3.3 it is sufficient to show that $(\mathsf{id} \oslash f) \circ (g \oslash \mathsf{id}) \circ \sigma_{AB} \sqsubseteq \sigma_{CD} \circ f$ i.e. $(g \oslash f) \circ \sigma_{AB} \sqsubseteq \sigma_{CD} \circ f$, which we know is true by lax naturality of σ.

The fact that $\mathbb{(}\ \eta\ \mathbb{)}$ is the unique morphism such that $\alpha \circ \mathbb{(}\ \eta\ \mathbb{)} = (\mathsf{id} \oslash \mathbb{(}\ \eta\ \mathbb{)}) \circ \eta$ inherits from this property in $\mathcal{G}$.

The commutative comonoid $!F \Rightarrow !F \otimes !F$ is defined pointwise from $\mathcal{G}$ with $(\mathsf{d}_F)(A) = \mathsf{d}_{F(A)}$. We know that this is a lax natural transformation by horizontal composition, since d is natural. The commutative comonoid equations inherit pointwise from $\mathcal{G}$, as does the equation $\mathsf{wk} \circ (\mathsf{der} \otimes \mathsf{id}) \circ \mathsf{d} = \alpha$.

Finally we need to check that for any commutative comonoid $m : F \Rightarrow F \otimes F$ and $g : G \Rightarrow F$, $g^\dagger = \mathbb{(}\ \mathsf{wk} \circ (g \otimes \mathsf{id}) \circ m\ \mathbb{)}$ is the unique comonoid morphism with $g = \mathsf{der} \circ g^\dagger$. Let m and g be as such. We can show $g^\dagger$ is a comonoid morphism pointwise, since each m_A is a commutative comonoid in $\mathcal{G}$. For uniqueness, we also note that any comonoid morphism $h : G \Rightarrow !F$ satisfying $g = \mathsf{der} \circ h$ at component A must be a comonoid morphism $G(A) \to !F(A)$ in $\mathcal{G}$ satisfying $g_A = \mathsf{der} \circ h_A$. Thus $h_A = !g_A$ by uniqueness in $\mathcal{G}$. Since this holds for all objects A, we have $h = g^\dagger$. ∎

Proposition 4.2.5 *$\mathcal{W}^{\mathcal{C}}$ is a WS!-category.*

Proof We proceed precisely as in Proposition 4.2.5, lifting the structure of a WS!-category in $\mathcal{W}$ to that in $\mathcal{W}^{\mathcal{C}}$. In particular, pointwise-winningness of the relevant morphisms in $\mathcal{W}^{\mathcal{C}}$ inherits from the winningness in $\mathcal{W}$. ∎

We can hence interpret WS! in these categories, for any $\mathcal{C}$. In particular, we see that each operation on winning strategies denoted by a proof rule lifts to an operation on uniform winning strategies.

Category of $\mathcal{L}$-structures

Definition Given a set of variables X and context Θ, we let $\mathcal{M}_\Theta^X$ denote the category of Θ-satisfying $\mathcal{L}$-models over X. Objects are $\mathcal{L}$-models over X that satisfy each formula in Θ. A morphism $(L, v) \to (L', v')$ is a map $f : |L| \to |L'|$ such that:

- For each $x \in X$, $v'(x) = f(v(x))$
- If $(L, v) \models \overline{\phi}(\overrightarrow{a})$ for $\overrightarrow{a} \in |L|^{\mathsf{ar}(\phi)}$ then $(L', v') \models \overline{\phi}(\overrightarrow{f(a)})$
- For each function symbol g in $\mathcal{L}$, $f(I_L(g)(\overrightarrow{a})) = I_{L'}(g)(\overrightarrow{f(a)})$.

Note that since the positive atoms include inequality, such morphisms must be injective. Also note that if $f : (L, v) \to (L', v')$ and $(L, v) \models \overline{\phi}(\overrightarrow{s})$ then $(L', v') \models \overline{\phi}(\overrightarrow{s})$.

If v is a valuation on X, define $v[x \mapsto l]$ on $X \cup \{x\}$ to be the valuation sending y to $v(y)$ if $y \neq x$, and x to l. Given $f : (L, v) \to (M, w)$ in $\mathcal{M}_X^\Theta$ and s a term

with $FV(s) \subseteq X$, f is also a map $(L, v[x \mapsto v(s)]) \to (M, w[x \mapsto w(s)])$ in $\mathcal{M}^{\Theta}_{X \cup \{x\}}$. We know that f preserves all of the valuations other than x, and for x we see that $f(v[x \mapsto v(s)](x)) = f(v(s)) = w(s) = w[x \mapsto w(s)](x)$.

We will give semantics of sequents $X; \Theta \vdash \Gamma$ as functors $\mathcal{M}^{X}_{\Theta} \to \mathcal{G}_e$, and proofs as uniform winning strategies. Proofs of $X; \Theta \vdash M, \Gamma$ will be given semantics as an arrow $\kappa_I \to [\![X; \Theta \vdash M, \Gamma]\!]$ and proofs of $X; \Theta \vdash P, \Gamma$ as an arrow $[\![X; \Theta \vdash P, \Gamma]\!] \to \kappa_\perp$.

4.2.2 Quantifiers as Adjoints

In this section, we will discuss an adjunction that will allow us to interpret the quantifiers. By Proposition 4.2.5, if $X; \Theta$ is fixed then $\mathcal{W}^{\mathcal{M}^{\Theta}_X}$ is a $\mathsf{WS!}$-category. This will thus give interpretation of all of the rules of $\mathsf{WS!}$. For the $\mathsf{P}_\forall$ and $\mathsf{P}_\exists$ rule, however, we must pass between $\mathcal{W}^{\mathcal{M}^{\Theta}_X}$ for varying X.

We first define some functors between these categories.

- If $FV(s) \subseteq X$ we can define a functor $\mathsf{set}^x_s : \mathcal{M}^{\Theta}_X \to \mathcal{M}^{\Theta}_{X \uplus \{x\}}$ by $\mathsf{set}^x_s(L, v) = (L, v[x \mapsto v(s)])$ and if $f : (L, v) \to (M, w)$ we set $\mathsf{set}^x_s(f) = f$. We need to check that $\mathsf{set}^x_s(f)$ is a valid morphism. We know that $\mathsf{set}^x_s(f)$ preserves all variables in X, and $\mathsf{set}^x_s(f)(v[x \mapsto v(s)](x)) = f(v(s)) = w(s) = w[x \mapsto w(s)](x)$ as required. It is clear that set^x_s is functorial.

  From this we can extract a functor $\mathsf{set}'^x_s : \mathcal{W}^{\mathcal{M}^{\Theta}_{X \uplus \{x\}}} \to \mathcal{W}^{\mathcal{M}^{\Theta}_X}$, mapping F to $F \circ \mathsf{set}^x_s$, with an action on arrows defined by horizontal composition.

- If x does not occur in Θ, there is an evident forgetful functor $U_x : \mathcal{M}^{\Theta}_{X \uplus \{x\}} \to \mathcal{M}^{\Theta}_X$ mapping (L, v) to $(L, v - x)$. From this we can extract a functor $U'_x : \mathcal{W}^{\mathcal{M}^{\Theta}_X} \to \mathcal{W}^{\mathcal{M}^{\Theta}_{X \uplus \{x\}}}$ mapping F to $F \circ U_x$, with an action on arrows defined by horizontal composition. Note that $U_x \circ \mathsf{set}^x_s = \mathsf{id}$ and so $\mathsf{set}'^x_s \circ U'_x = \mathsf{id}$.

We will show that U'_x has a right adjoint $\forall x.\_$. Assuming empty Γ, this allows us to interpret the rules $\mathsf{P}_\forall$ and $\mathsf{P}_\exists$.

- For $\mathsf{P}_\forall$, the premise is a map $I \to \forall x.[\![N]\!]$ in $\mathcal{W}^{\mathcal{M}^{\Theta}_{X \uplus \{x\}}}$ and the conclusion is a map $I = U'_x(I) \to [\![N]\!]$ in $\mathcal{W}^{\mathcal{M}^{\Theta}_X}$. The adjunction assures us that there is a natural bijection between this hom sets.

$$\mathsf{P}_\forall \frac{X \uplus \{x\}; \Theta \vdash \forall x.N}{X; \Theta \vdash \forall x.N}$$

- The premise of the $\mathsf{P}_\exists$ rule provides a map $[\![P[s/x]]\!] \to \perp$. Since $[\![P[s/x]]\!] = \mathsf{set}'^x_s([\![P]\!])$, and the conclusion requires map $\forall x.[\![P]\!] \to \perp$, we need only provide a map $\forall x.[\![P]\!] \to \mathsf{set}'^x_s([\![P]\!])$.

$$\mathsf{P}^s_\exists \frac{X;\Theta \vdash P[s/x]}{X;\Theta \vdash \exists x.P}\, FV(s) \subseteq X$$

We note that the adjunction comes equipped with a unit $U'_x \circ \forall x \Rightarrow \mathsf{id}$. This yields a map $\forall x.[\![P]\!] = \mathsf{set}'^x_y(U'_x(\forall x.[\![P]\!])) \Rightarrow \mathsf{set}'^x_y([\![P]\!])$ as required.

We can hence use the existence of a right adjoint for U'_x (together with a suitable distributivity isomorphism to deal with nonempty Γ) to give semantics to $\mathsf{P}_\forall$ and $\mathsf{P}_\exists$. We next show that U'_x has a right adjoint $\forall x.\_$.

The Famlnj Construction

Definition Let $\mathcal{C}$ be a category. We define the category $\mathsf{Famlnj}(\mathcal{C})$. An object is a set I and a family of $\mathcal{C}$-objects $\{A_i : i \in I\}$. An arrow $\{A_i : i \in I\} \to \{B_j : j \in J\}$ is a pair $(f, \{f_i : i \in I\})$ where f is an injective function $I \to J$ and each $f_i : A_i \to B_{f(i)}$. We will often write such a map as $(f, \{f_i\})$ when we wish to leave the indexing set implicit.

- Composition is defined by $(f, \{f_i\}) \circ (g, \{g_i\}) = (f \circ g, \{f_{g(i)} \circ g_i\})$.
- The identity $\{A_i : i \in I\} \to \{A_i : i \in I\}$ is given by $(\mathsf{id}, \{\mathsf{id}_{A_i}\})$.
- Satisfaction of the categorical axioms is inherited from $\mathcal{C}$.

Definition Let $F : \mathcal{C} \to \mathcal{D}$. We define $\mathsf{Famlnj}(F) : \mathsf{Famlnj}(\mathcal{C}) \to \mathsf{Famlnj}(\mathcal{D})$. On objects, $\mathsf{Famlnj}(F)(\{A_i : i \in I\}) = \{F(A_i) : i \in I\}$. On arrows, $\mathsf{Famlnj}(F)(f, \{f_i\}) = (f, \{F(f_i)\})$. We need to show that $\mathsf{Famlnj}(F)$ is a functor:

- It is clear that $\mathsf{Famlnj}(F)$ preserves the identity.
- For composition: $\mathsf{Famlnj}(F)((f, \{f_i\}) \circ (g, \{g_i\})) = \mathsf{Famlnj}(F)(f \circ g, \{f_{g(i)} \circ g_i\}) = (f \circ g, F(f_{g(i)} \circ g_i)) = (f \circ g, F(f_{g(i)}) \circ F(g_i)) = (f, \{F(f_i)\}) \circ (g, \{F(g_i)\}) = \mathsf{Famlnj}(F)(f, \{f_i\}) \circ \mathsf{Famlnj}(F)(g, \{g_i\})$.

We define a distributivity functor $\mathsf{dst} : \mathsf{Famlnj}(\mathcal{C}) \times \mathcal{D} \to \mathsf{Famlnj}(\mathcal{C} \times \mathcal{D})$ by $\mathsf{dst}(\{A_i : i \in I\}, B) = \{(A_i, B) : i \in I\}$ and $\mathsf{dst}((f, \{f_i\}), g) = (f, \{(f_i, g)\})$.

Constructing $\forall x.F$

First, we construct arbitrary products on games.

Definition Let X be a set and $\{A_x : x \in X\}$ a family of win-games indexed by X, each with polarity p. We define the game $S = \prod_{x \in X} A_x$ by $(M_S, \Lambda_S, p, P_S, W_S)$ by:

- $M_S = \sum_{x \in X} M_{A_x}$
- $\lambda_S(\mathsf{in}_x(m)) = \lambda_{A_x}(m)$

- $P_S = \{\mathsf{in}^*_x(s) : x \in X, s \in P_{A_x}\}$
- $W_s = \{\mathsf{in}^\omega_x(s) : x \in X, s \in W_{A_x}\}$

If $p = O$ then this is the product over an X-indexed family of games, if $p = P$ then this acts as the coproduct. We can thus equip the game categories $\mathcal{G}$, $\mathcal{G}_e$, $\mathcal{W}_s$ and $\mathcal{W}$ with arbitrary products.

Suppose F is an object in $\mathcal{W}^{\mathcal{M}^\Theta_{X \uplus \{x\}}}$ (a functor $\mathcal{M}^\Theta_{X \uplus \{x\}} \to \mathcal{G}_e$). We define $\forall x.F$ as an object in $\mathcal{W}^{\mathcal{M}^\Theta_X}$ (a functor $\mathcal{M}^\Theta_X \to \mathcal{G}_e$).

We first define a product functor $\mathsf{prod} : \mathsf{FamInj}(\mathcal{G}_e) \to \mathcal{G}_e$. On objects, prod sends $\{G_i : i \in I\}$ to $\prod_{i \in I} G_i$. On arrows, let $f : \{G_j : j \in J\} \to \{H_h : h \in H\}$. The embedding part of $\mathsf{prod}(f)$ is given by $\langle g_h \rangle_h$ where $g_h = i_{f_j} \circ \pi_j$ if $h = f(j)$ and ϵ otherwise. The projection part is given by $\langle p_{f_j} \circ \pi_{f(j)} \rangle_j$. We must check that:

- $\mathsf{prod}(f)$ **is a valid embedding-projection pair:** We need to check that $p_{\mathsf{prod}(f)} \circ i_{\mathsf{prod}(f)} = \mathsf{id}$. If $f : \{A_i : i \in I\} \to \{B_j : j \in J\}$ we need to show that $\pi_i \circ p_{\mathsf{prod}(f)} \circ i_{\mathsf{prod}(f)} = \pi_i$ for each i. But this is $p_{f_i} \circ \pi_{f(i)} \circ i_{\mathsf{prod}(f)} = p_{f_i} \circ i_{f_i} \circ \pi_i = \pi_i$ as required. We also need to check that $\pi_j \circ i_{\mathsf{prod}(f)} \circ p_{\mathsf{prod}(f)} \sqsubseteq \pi_j$ for each j. If j is not in the image of f, then $\pi_j \circ i_{\mathsf{prod}(f)} \circ p_{\mathsf{prod}(f)} = \epsilon$ and so we are done. If $j = f(i)$ then $\pi_j \circ i_{\mathsf{prod}(f)} \circ p_{\mathsf{prod}(f)} = i_{f_i} \circ \pi_i \circ p_{\mathsf{prod}(f)} = i_{f_i} \circ p_{f_i} \circ \pi_i \sqsubseteq \mathsf{id} \circ \pi_{f(i)} = \pi_{f(i)} = \pi_j$ as required.
- prod **preserves the identity:** To see that $\mathsf{prod}(\mathsf{id}) = \mathsf{id}$, note that $\mathsf{prod}(\mathsf{id}, \{(\mathsf{id}, \mathsf{id})\}) = (\langle \mathsf{id} \circ \pi_j \rangle_j, \langle \mathsf{id} \circ \pi_i \rangle_i) = (\mathsf{id}, \mathsf{id}) = \mathsf{id}$.
- prod **preserves composition:** Suppose $g : \{A_j : j \in J\} \to \{B_k : k \in K\}$ and $f : \{B_k : k \in K\} \to \{C_l : l \in L\}$. We need to show that $i_{\mathsf{prod}(f \circ g)} = i_{\mathsf{prod}(f)} \circ i_{\mathsf{prod}(g)}$ and $p_{\mathsf{prod}(f \circ g)} = p_{\mathsf{prod}(g)} \circ p_{\mathsf{prod}(f)}$.

  We first show that $i_{\mathsf{prod}(f \circ g)} = i_{\mathsf{prod}(f)} \circ i_{\mathsf{prod}(g)}$. We show that for each l, $\pi_l \circ i_{\mathsf{prod}(f \circ g)} = \pi_l \circ i_{\mathsf{prod}(f)} \circ i_{\mathsf{prod}(g)}$. We consider cases.
  - If $l = f(g(j))$ then LHS is $i_{f_{g(j)} \circ g_j} \circ \pi_j = i_{f_{g(j))}} \circ i_{g_j} \circ \pi_j$. The RHS is $\pi_{f(g(j))} \circ i_{\mathsf{prod}(f)} \circ i_{\mathsf{prod}(g)} = i_{f_{g(j)}} \circ \pi_{g(j)} \circ i_{\mathsf{prod}(g)} = i_{f_{g(j))}} \circ i_{g_j} \circ \pi_j$ as required.
  - If l is not in the range of $f \circ g$ then either l is not in the range of f, or $l = f(k)$ and k is not in the range of g. In the first case, the LHS is ϵ and the RHS is $\epsilon \circ i_{\mathsf{prod}(g)} = \epsilon$, as required.
  - In the second case, if $l = f(k)$ and k is not in the range of g, the LHS is ϵ and the RHS is $i_{f_k} \circ \pi_k \circ i_{\mathsf{prod}}(g) = i_{f_k} \circ \epsilon$. This is ϵ since i_{f_k} is strict, as required.

  We next show that $p_{\mathsf{prod}(f \circ g)} = p_{\mathsf{prod}(g)} \circ p_{\mathsf{prod}(f)}$. We show that for each j, $\pi_j \circ p_{\mathsf{prod}(f \circ g)} = \pi_j \circ p_{\mathsf{prod}(g)} \circ p_{\mathsf{prod}(f)}$. Well the LHS is $p_{f_{g(j)} \circ g_j} \circ \pi_{f(g(j))} = p_{g_j} \circ p_{f_{g(j)}} \circ \pi_{f(g(j))}$. The RHS is $p_{g_j} \circ \pi_{g(j)} \circ p_{\mathsf{prod}(f)} = p_{g_j} \circ p_{f_{g(j)}} \circ \pi_{f(g(j))}$ as required.

We next define a functor $\mathsf{add}_x : \mathcal{M}_X^\Theta \to \mathsf{FamInj}(\mathcal{M}_{X \uplus \{x\}}^\Theta)$ which sends a model (L, v) to the family $\{(L, v[x \mapsto i]) : i \in |L|\}$. On arrows, if $f : (L, v) \to (L', v')$ we set $\mathsf{add}_x(f) = (f, \{f_i\})$ where $f_i : (L, v[x \mapsto i]) \to (L', v[x \mapsto f(i)])$ is just f. We need to check that this is functorial.

- It is clear that the identity is preserved.
- For composition, $\mathsf{add}_x(f \circ g) = (f \circ g, \{f \circ g\}) = (f, \{f\}) \circ (g, \{g\}) = \mathsf{add}_x(f) \circ \mathsf{add}_x(g)$.

Finally, given $F : \mathcal{M}_{X \uplus \{x\}}^\Theta \to \mathcal{G}_e$ we define $\forall x.F : \mathcal{M}_X^\Theta \to \mathcal{G}_e$ to be $\mathsf{prod} \circ \mathsf{FamInj}(F) \circ \mathsf{add}_x$.

Constructing the Unit

We must next give the unit of this adjunction. For each F, we must give a uniform winning strategy $\eta : U'_x(\forall x.F) \Rightarrow F$. Such an η is a winning uniform strategy $\mathsf{prod} \circ \mathsf{FamInj}(F) \circ \mathsf{add}_x \circ U_x \Rightarrow F$. Note that $(\mathsf{prod} \circ \mathsf{FamInj}(F) \circ \mathsf{add}_x \circ U_x)(L, v) = \mathsf{prod}(\{F(L, v - x[x \mapsto l]) : l \in L\}) = \prod_{l \in L} F(L, v[x \mapsto l])$. Thus $\eta_{(L,v)}$ must be a winning strategy $\prod_{l \in L} F(L, v[x \mapsto l]) \to F(L, v)$ and we take $\eta_{(L,v)} = \pi_{v(x)}$.

For uniformity, the following diagram must lax-commute:

$$\begin{array}{ccc} \prod_{l \in L} F(L, v[x \mapsto l]) & \xrightarrow{\pi_{v(x)}} & F(L, v[x \mapsto v(s)]) \\ \Big\downarrow{\scriptstyle i_{\mathsf{prod}(F(f))}} & & \Big\downarrow{\scriptstyle i_{F(f)}} \\ \prod_{m \in M} F(M, w[x \mapsto m]) & \xrightarrow[\pi_{w(x)}]{} & F(M, w[x \mapsto w(s)]) \end{array}$$

Note that $w(x) = f(v(x))$ and $i_{\mathsf{prod}(F)(f)} = \langle g_n \rangle_n$ where $g_{f(y)} = i_{F(f)} \circ \pi_y$. Thus, $\pi_{w(x)} \circ i_{\mathsf{prod}(F)(f)} = \pi_{f(v(x))} \circ i_{\mathsf{prod}(F)(f)} = i_{F(f)} \circ \pi_{v(x)}$ as required.

Universal Property

Given $f : U'_x(F) \to G$ we must show that there is a unique $\hat{f} : F \to \forall x.G$ such that $f = \eta_G \circ U'_x(\hat{f})$. Let f be such a uniform winning strategy. Then we must give winning strategies $\hat{f}_{(L,v)} : F(L, v) \to \prod_{l \in L} G(L, v[x \mapsto l])$. Set $\hat{f}_{(L,v)} = \langle h_l \rangle_l$ where $h_l : F(L, v) \to G(L, v[x \mapsto l])$ is defined by $f_{(L, v[x \mapsto l])}$. To see that $\hat{f}$ is uniform, the

following diagram must lax-commute:

$$\begin{array}{ccc} F(L,v) & \xrightarrow{\hat{f}_{(L,v)}} & \prod_{l\in L} G(L, v[x \mapsto l]) \\ \Big\downarrow{\scriptstyle i_{F(f)}} & \sqsupseteq & \Big\downarrow{\scriptstyle i_{\mathsf{prod}(G(f))}} \\ F(M,w) & \xrightarrow{\hat{f}_{(M,w)}} & \prod_{m\in M} G(M, w[x \mapsto m]) \end{array}$$

It is sufficient to show that $\pi_m \circ \hat{f}_{(M,w)} \circ i_{F(f)} \sqsupseteq \pi_m \circ i_{\mathsf{prod}(G(f))} \circ \hat{f}_{(L,v)}$ for each m. If m is not in the image of f, the right-hand side is $\{\epsilon\}$ and so we are done. If $m = f(j)$ the RHS is $i_{G(f)} \circ \pi_j \circ \hat{f}_{(L,v)} = i_{G(f)} \circ f_{(L,v[x\mapsto j])}$ and the LHS is $f_{(M,w[x\mapsto m])} \circ i_{F(f)} = f_{(M,w[x\mapsto f(j)])} \circ i_{F(f)}$. Thus it is sufficient to show that $f_{(M,w[x\mapsto f(j)])} \circ i_{F(f)} \sqsupseteq i_{G(f)} \circ f_{(L,v[x\mapsto j])}$, which follows from lax naturality of f.

We next need to show that $\hat{f}$ satisfies the universal property. First, we must show that $f = \eta_G \circ U'_x(\hat{f})$. It suffices to show that for each (L,v), $f_{(L,v)} = ((\eta_G) \circ U'_x(\hat{f}))_{L,v}$. Composition in is given by vertical composition. Thus, the RHS is given by $\pi_{v(x)} \circ \langle f_{(L,v[x\mapsto l])} \rangle_l = f_{(L,v[x\mapsto v(x)])} = f_{(L,v)}$ as required.

We need to show that $\hat{f} : F \to \forall x.G$ is the *unique* uniform strategy satisfying $f = \eta_G \circ U'_x(\hat{f})$. Suppose $h : F \to \forall x.G$ in $\mathcal{W}^{\mathcal{M}^\Theta_X}$ satisfies this property. Then given (L,v) in $\mathcal{M}^\Theta_{X \uplus \{x\}}$, we know that $f_{(L,v)} = \eta_{G(L,v)} \circ h_{(L,v-x)} = \pi_{v(x)} \circ h_{(L,v-x)}$. (1)

Let $(L,v) \in \mathcal{M}^\Theta_X$. We must show that $h_{(L,v)} = \hat{f}_{(L,v)} = \langle f_{(L,v[x\mapsto l])} \rangle_l$. Thus we need to show that for each l, $\pi_l \circ h_{(L,v)} = f_{(L,v[x\mapsto l])}$. But consider the model $(L, v[x \mapsto l])$. Then by (1) $f_{(L,v[x\mapsto l])} = \pi_{v[x\mapsto l](x)} \circ h_{(L,v[x\mapsto l]-x)} = \pi_l \circ h_{(L,v)}$, as required.

Thus, we see that

Proposition 4.2.6 *The functor $U'_x : \mathcal{W}^{\mathcal{M}^\Theta_X} \to \mathcal{W}^{\mathcal{M}^\Theta_{X\uplus\{x\}}}$ has a right adjoint given by $\forall x.\_ = \mathsf{prod} \circ \mathsf{FamInj}(\_) \circ \mathsf{add}_x : \mathcal{W}^{\mathcal{M}^\Theta_{X\uplus\{x\}}} \to \mathcal{W}^{\mathcal{M}^\Theta_X}$*

Concretely, if $N : \mathcal{M}^\Theta_{X\uplus\{x\}} \to \mathcal{G}_e$ then on objects $[\![\forall x.N]\!](L,v) = \prod_{l\in|L|} [\![N]\!](L, v[x \mapsto l])$. On arrows, if $f : (L,v) \to (L',w)$ then $[\![\forall x.N]\!](f) : \prod_{l\in|L|} [\![N]\!](L, v[x \mapsto l]) \to \prod_{l\in|L'|} [\![N]\!](L', w[x \mapsto l])$ is given as follows: The embedding part (left to right) is given by $\langle g_m \rangle_m$ where $g_m = \epsilon$ if m is not in the image of f, and $g_m = i[\![N]\!](f) \circ \pi_l$ if $m = f(l)$ (note in this case l is unique by injectivity of f). The projection part is given by $\langle p[\![N]\!](f) \circ \pi_{f(l)} \rangle_l$.

Family of Adjunctions

We have exhibited an adjunction $U'_x \dashv \forall x.\_$ for each X and x, but we have not discussed how these adjunctions fit together. Here we consider an indexed category $\mathcal{W}^{\mathcal{M}_X}$ over the variable set X, and show that the family of adjunctions satisfy a Beck-Chevalley condition. In this section we assume that the Θ component is empty.

Let **Set** denote the category of (variable) sets and functions between them. We can describe a functor $I : \mathbf{Set} \to \mathbf{Cat}$.

- On objects, $I(X) = \mathcal{W}^{\mathcal{M}_X}$.
- On arrows, if $f : X \to Y$ we must give a functor $I(f)(F) : \mathcal{W}^{\mathcal{M}_X} \to \mathcal{W}^{\mathcal{M}_Y}$.
  - Define $\tilde{f} : \mathcal{M}_Y \to \mathcal{M}_X$. On objects, $\tilde{f}(M, v) = (M, v \circ f)$. If $h : (M, v) \to (N, v')$ in $\mathcal{M}_Y$, then $h \circ v = v'$ and so $h \circ v \circ f = v' \circ f$ and so $h : (M, v \circ f) \to (N, v' \circ f)$ in $\mathcal{M}_X$, let $\tilde{f}(h) = h$.
  - On objects, if $F : \mathcal{M}_X \to \mathcal{G}_e$ we define $I(f)(F) : \mathcal{M}_Y \to \mathcal{G}_e = F \circ \tilde{f}$.
  - On arrows, if $\sigma : F \Rightarrow G : \mathcal{M}_X \to \mathcal{G}_e$, define $I(f)(F)(\sigma) : F \circ \tilde{f} \Rightarrow G \circ \tilde{f} = \sigma\tilde{f}$ using horizontal composition.

Note that $U'_x : \mathcal{W}^{\mathcal{M}}_X \to \mathcal{W}^{\mathcal{M}}_{X \uplus \{x\}} = I(\mathsf{in}_1)$ and the following diagram commutes:

$$\begin{array}{ccc} \mathcal{W}^{\mathcal{M}_X} & \xrightarrow{I(\mathsf{in}_1)} & \mathcal{W}^{\mathcal{M}_{X \uplus \{x\}}} \\ {\scriptstyle I(f)}\downarrow & & \downarrow{\scriptstyle I(f \uplus \{x \mapsto y\})} \\ \mathcal{W}^{\mathcal{M}_Y} & \xrightarrow{I(\mathsf{in}_1)} & \mathcal{W}^{\mathcal{M}_{Y \uplus \{y\}}} \end{array}$$

Since $I(\mathsf{in}_1) \dashv \forall x.\_$, the Beck-Chevalley condition asks if the following diagram commutes:

$$\begin{array}{ccc} \mathcal{W}^{\mathcal{M}_X} & \xleftarrow{\forall x.\_} & \mathcal{W}^{\mathcal{M}_{X \uplus \{x\}}} \\ {\scriptstyle I(f)}\downarrow & & \downarrow{\scriptstyle I(f \uplus \{x \mapsto y\})} \\ \mathcal{W}^{\mathcal{M}_Y} & \xleftarrow{\forall y.\_} & \mathcal{W}^{\mathcal{M}_{Y \uplus \{y\}}} \end{array}$$

We check this extensionally, first considering objects. Let $F \in \mathcal{M}_{X \uplus \{x\}}$ and $(M, v) \in \mathcal{M}_Y$. Then the LHS at (M, v) is $(\forall x.F)(M, v \circ f) = \prod_{m \in M} F(M, v \circ f[x \mapsto m])$. The RHS is $[\forall y.I(f \uplus \{x \mapsto y\})(F)](M, v) = \prod_{m \in M} I(f \uplus \{x \mapsto y\})(F)(M, v[y \mapsto m]) = \prod_{m \in M} F(M, v[y \mapsto m] \circ (f \uplus \{x \mapsto y\})) = \prod_{m \in M} F(M, v \circ f[x \mapsto m])$ as required.

On arrows, let $\sigma : F \Rightarrow G : \mathcal{M}_{X \uplus \{x\}} \to \mathcal{G}_e$ and $(M, v) \in \mathcal{M}_Y$. We must check that $I(f)(\forall x.\sigma)_{(M,v)} = (\forall y.I(f \uplus \{x \mapsto y\})(\sigma))_{(M,v)}$. But $\forall x.\sigma = \widehat{\sigma \circ \eta}$ and so the LHS is $\widehat{\sigma \circ \eta}_{(M,v \circ f)} = \langle (\sigma \circ \eta)_{(M,v \circ f[x \mapsto l])} \rangle_l = \langle \sigma_{(M,v \circ f[x \mapsto l])} \circ \pi_l \rangle_l$. The RHS is $(\forall y.I(f \uplus \{x \mapsto y\})(\sigma))_{(M,v)} = (I(f \uplus \{\widehat{x \mapsto y}\})(\sigma) \circ \eta)_{(M,v)} = \langle (I(f \uplus \{x \mapsto y\})(\sigma) \circ \eta)_{(M,v[y \mapsto l])} \rangle_l = \langle \sigma_{(M,v[y \mapsto l] \circ (f \uplus \{x \mapsto y\}))} \circ \pi_l \rangle_l = \langle \sigma_{(M,v \circ f[x \mapsto l])} \circ \pi_l \rangle_l$ as required.

4.2.3 Semantics of Sequents

We next give semantics of sequents $X; \Theta \vdash \Gamma$ as functors $\mathcal{M}_X^\Theta \to \mathcal{G}_e$. For the formulas of WS!, we note that $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a WS!-category, and so we can use the categorical semantics of sequents therein. We exhibit these concretely here. For brevity, Φ ranges over $X; \Theta$ contexts. If $F, G : \mathcal{M}_X^\Theta \to \mathcal{G}_e$ and $\odot : \mathcal{G}_e \times \mathcal{G}_e \to \mathcal{G}_e$ then we write $F \odot G : \mathcal{M}_X^\Theta \to \mathcal{G}_e$ for $\odot \circ (F \times G) \circ \Delta$, where $\Delta : \mathcal{C} \to \mathcal{C} \times \mathcal{C}$ is the diagonal functor.

$$
\begin{array}{lcllcl}
[\![\Phi \vdash \mathbf{1}]\!] & = & \kappa_I & [\![\Phi \vdash \mathbf{0}]\!] & = & \kappa_I \\
[\![\Phi \vdash \bot]\!] & = & \kappa_\bot & [\![\Phi \vdash \top]\!] & = & \kappa_\bot \\
[\![\Phi \vdash M \otimes N]\!] & = & [\![\Phi \vdash M]\!] \otimes [\![\Phi \vdash N]\!] & [\![\Phi \vdash P \parr Q]\!] & = & [\![\Phi \vdash P]\!] \otimes [\![\Phi \vdash Q]\!] \\
[\![\Phi \vdash M \& N]\!] & = & [\![\Phi \vdash M]\!] \times [\![\Phi \vdash N]\!] & [\![\Phi \vdash P \oplus Q]\!] & = & [\![\Phi \vdash P]\!] \times [\![\Phi \vdash Q]\!] \\
[\![\Phi \vdash M \oslash N]\!] & = & [\![\Phi \vdash M]\!] \oslash [\![\Phi \vdash N]\!] & [\![\Phi \vdash P \lhd Q]\!] & = & [\![\Phi \vdash P]\!] \oslash [\![\Phi \vdash Q]\!] \\
[\![\Phi \vdash M \lhd Q]\!] & = & [\![\Phi \vdash Q]\!] \multimap [\![\Phi \vdash M]\!] & [\![\Phi \vdash P \oslash N]\!] & = & [\![\Phi \vdash N]\!] \multimap [\![\Phi \vdash P]\!] \\
[\![\Phi \vdash !N]\!] & = & ![\![\Phi \vdash N]\!] & [\![\Phi \vdash ?P]\!] & = & ![\![\Phi \vdash P]\!] \\
[\![\Phi \vdash \phi(\overrightarrow{s})]\!](L, v) & = & I \text{ if } (L, v) \models \overline{\phi}(\overrightarrow{s}) & [\![\Phi \vdash \overline{\phi}(\overrightarrow{s})]\!](L, v) & = & I \text{ if } (L, v) \models \overline{\phi}(\overrightarrow{s}) \\
[\![\Phi \vdash \phi(\overrightarrow{s})]\!](L, v) & = & \bot \text{ if } (L, v) \models \phi(\overrightarrow{s}) & [\![\Phi \vdash \overline{\phi}(\overrightarrow{s})]\!](L, v) & = & \bot \text{ if } (L, v) \models \phi(\overrightarrow{s})
\end{array}
$$

$$
\begin{array}{lcl}
[\![X; \Theta \vdash \forall x.N]\!] & = & \forall x.[\![X \uplus \{x\}; \Theta \vdash N]\!] \\
[\![X; \Theta \vdash \exists x.P]\!] & = & \forall x.[\![X \uplus \{x\}; \Theta \vdash P]\!]
\end{array}
$$

$$
\begin{array}{lcllcl}
[\![\Phi \vdash M, \Gamma, N]\!] & = & [\![\Phi \vdash M, \Gamma]\!] \oslash [\![\Phi \vdash N]\!] & [\![\Phi \vdash M, \Gamma, P]\!] & = & [\![\Phi \vdash P]\!] \multimap [\![\Phi \vdash M, \Gamma]\!] \\
[\![\Phi \vdash P, \Gamma, N]\!] & = & [\![\Phi \vdash N]\!] \multimap [\![\Phi \vdash P, \Gamma]\!] & [\![\Phi \vdash P, \Gamma, Q]\!] & = & [\![\Phi \vdash P, \Gamma]\!] \oslash [\![\Phi \vdash Q]\!]
\end{array}
$$

In the case of atoms, the functors are specified pointwise on objects, and we must also define the (functorial) action on arrows. Let $f : (L, v) \to (L', v')$. If the truth value of $\phi(\overrightarrow{s})$ is the same in (L, v) and (L', v), we use the identity embedding $(\mathsf{id}, \mathsf{id})$. If the truth value of $\phi(\overrightarrow{s})$ is different, we must have $(L, v) \models \phi(\overrightarrow{s})$ and $(L', v) \models \overline{\phi}(\overrightarrow{s})$ since morphisms in $\mathcal{M}_X$ preserve truth of positive atoms. Thus we need an embedding $I \to \bot$. We can take $(\epsilon_{I \multimap \bot}, \epsilon_{\bot \multimap I})$ where ϵ_A is the strategy containing just the empty sequence. Note that $\epsilon_{\bot \multimap I} \circ \epsilon_{I \multimap \bot} = \epsilon_I = \mathsf{id}_I$ and $\epsilon_{I \multimap \bot} \circ \epsilon_{\bot \multimap I} = \epsilon \sqsubseteq \mathsf{id}_\bot$ (ϵ is the

bottom element with respect to $\sqsubseteq$).

We must check functoriality. We have already noted that if the truth value of $\phi(\overrightarrow{s})$ is the same in (L, v) and (L', v') then $[\![\phi(\overrightarrow{s})]\!](f) = \mathsf{id}$, so in particular $[\![\phi(\overrightarrow{s})]\!](\mathsf{id}) = \mathsf{id}$. For composition, suppose $f : (L, v) \to (L', v')$ and $g : (L', v') \to (L'', v'')$. We can consider the truth value of $\phi(\overrightarrow{s})$ in each of these models (only some cases are possible, as morphisms preserve truth of positive atoms).

$(L, v) \models$	$(L', v') \models$	$(L'', v'') \models$	$[\![\phi(\overrightarrow{s})]\!](g) \circ [\![\phi(\overrightarrow{s})]\!](g) = [\![\phi(\overrightarrow{s})]\!](g \circ f)$
$\phi(\overrightarrow{s})$	$\phi(\overrightarrow{s})$	$\phi(\overrightarrow{s})$	$(\mathsf{id}, \mathsf{id}) \circ (\mathsf{id}, \mathsf{id}) = (\mathsf{id}, \mathsf{id})$
$\phi(\overrightarrow{s})$	$\phi(\overrightarrow{s})$	$\overline{\phi}(\overrightarrow{s})$	$(\epsilon, \epsilon) \circ (\mathsf{id}, \mathsf{id}) = (\epsilon, \epsilon)$
$\phi(\overrightarrow{s})$	$\overline{\phi}(\overrightarrow{s})$	$\overline{\phi}(\overrightarrow{s})$	$(\mathsf{id}, \mathsf{id}) \circ (\epsilon, \epsilon) = (\epsilon, \epsilon)$
$\overline{\phi}(\overrightarrow{s})$	$\overline{\phi}(\overrightarrow{s})$	$\overline{\phi}(\overrightarrow{s})$	$(\mathsf{id}, \mathsf{id}) \circ (\mathsf{id}, \mathsf{id}) = (\mathsf{id}, \mathsf{id})$

4.2.4 Contexts and Distributivity

In Section 2.3.4 we noted that we can give an equivalent semantics of sequents via context-functors: each context Γ yields two endofunctors $[\![\Gamma]\!]^+, [\![\Gamma]\!]^- : \mathcal{W}_s^{\mathcal{M}_X^\Theta} \to \mathcal{W}_s^{\mathcal{M}_X^\Theta}$. These yield functors $[\![\Gamma]\!]_1^b : \mathcal{G}_s \times \mathcal{M}_X^\Theta \to \mathcal{G}_s$ given concretely by:

$$\begin{array}{lcllcl} [\![\epsilon]\!]_1^+ & = & \pi_1 & [\![\epsilon]\!]_1^- & = & \pi_1 \\ [\![\Gamma, P]\!]_1^+ & = & [\![\Gamma]\!]_1^+ \oslash (i \circ [\![P]\!] \circ \pi_2) & [\![\Gamma, P]\!]_1^- & = & (p \circ [\![P]\!] \circ \pi_2) \multimap [\![\Gamma]\!]_1^- \\ [\![\Gamma, M]\!]_1^+ & = & (p \circ [\![M]\!] \circ \pi_2) \multimap [\![\Gamma]\!]_1^+ & [\![\Gamma, M]\!]_1^- & = & [\![\Gamma]\!]_1^- \oslash (i \circ [\![M]\!] \circ \pi_2) \end{array}$$

Proposition 4.2.7 *If A is negative then $i \circ [\![A, \Gamma]\!] = [\![\Gamma]\!]_1^- \circ (i \circ [\![A]\!] \times \mathsf{id}) \circ \Delta$; if A is positive then $i \circ [\![A, \Gamma]\!] = [\![\Gamma]\!]_1^+ \circ (i \circ [\![A]\!] \times \mathsf{id}) \circ \Delta$.*

Proof Simple induction on Γ. ∎

We next show that if $x \notin FV(\Gamma)$ there is an isomorphism $\mathsf{dist}_\Gamma : [\![\forall x.A, \Gamma]\!] \cong \forall x.[\![A, \Gamma]\!]$ in $\mathcal{W}^{\mathcal{M}_X^\Theta}$.

First, we note that there is a natural isomorphism

$$\mathsf{dist}_\oslash : \_ \oslash \_ \circ (\mathsf{prod} \times \mathsf{id}) \Rightarrow \mathsf{prod} \circ \mathsf{FamInj}(\_ \oslash \_) \circ \mathsf{dst} : \mathsf{FamInj}(\mathcal{G}_s) \times \mathcal{G}_s \to \mathcal{G}_s$$

which concretely is a family of winning strategies

$$\mathsf{prod}(\{G_i : i \in I\}) \oslash M \to \mathsf{prod}(\{G_i \oslash M : i \in I\})$$

given by $\mathsf{dist}_\oslash = \langle \pi_i \oslash \mathsf{id} \rangle_i$. Each $\mathsf{dist}_\oslash$ is an is an isomorphism in $\mathcal{W}_s$.

We next check that $\mathsf{dist}_{\oslash}$ is a natural. If $(f, \{f_i\}) : \{G_i\} \to \{H_j\}$ in $\mathsf{FamInj}(\mathcal{G}_s)$ and $g : M \multimap N$ we have $\langle a_j \rangle_j = \mathsf{dist}_{\oslash} \circ \mathsf{prod}(f, \{f_i\}) \oslash g = \mathsf{prod}(f, \{f_i \oslash g\}) \circ \mathsf{dist}_{\oslash} = \langle b_j \rangle_j$. We show that $a_j = b_j$.

If j is not in the range of f, then $a_j = \pi_j \circ \mathsf{dist}_{\oslash} \circ \epsilon \oslash g = \pi_j \circ \mathsf{dist}_{\oslash} \circ \epsilon = \epsilon = \epsilon \circ \mathsf{dist}_{\oslash} = b_j$, as required. If $j = f(i)$ then $a_j = \pi_j \circ \mathsf{dist}_{\oslash} \circ \mathsf{prod}(f, \{f_i\}) \oslash g = (\pi_j \oslash \mathsf{id}) \circ (\mathsf{prod}(f, \{f_i\}) \oslash g) = (\pi_j \circ \mathsf{prod}(f, \{f_i\}) \oslash g) = f_i \circ \pi_i \oslash g$. While $b_j = (f_i \oslash g) \circ \pi_i \circ \mathsf{dist}_{\oslash} = (f_i \oslash g) \circ (\pi_i \oslash \mathsf{id}) = f_i \circ \pi_i \oslash g = a_j$ as required.

Similarly, we can define a natural isomorphism

$$\mathsf{dist}_{\multimap} : \mathsf{prod}(\{M \multimap G_i : i \in I\}) \cong M \multimap \mathsf{prod}(\{G_i : i \in I\})$$

between functors

$$\_ \multimap \_ \circ (\mathsf{prod} \times \mathsf{id}) \Rightarrow \mathsf{prod} \circ \mathsf{FamInj}(\_ \multimap \_) \circ \mathsf{dst} : \mathsf{FamInj}(\mathcal{G}_s) \times \mathcal{G}_s \to \mathcal{G}_s.$$

For each Γ, we can then construct a natural isomorphism

$$\mathsf{dist}_{b,\Gamma} : [\![\Gamma]\!]^b_1 \circ (\mathsf{prod} \times \mathsf{id}) \cong \mathsf{prod} \circ \mathsf{FamInj}([\![\Gamma]\!]^b_1) \circ \mathsf{dst} : \mathsf{FamInj}(\mathcal{G}_s) \times \mathcal{M}^{\Theta}_X \to \mathcal{G}_s$$

proceeding by induction on Γ:

- If $b = -$ and $\Gamma = \epsilon$ then we require a natural isomorphism $\mathsf{prod} \circ \pi_1 \cong \mathsf{prod} \circ \mathsf{FamInj}(\pi_1) \circ \mathsf{dst} : \mathsf{FamInj}(\mathcal{G}_s) \times \mathcal{M}^{\Theta}_X \to \mathcal{G}_s$. But it is clear that these functors are identical.
- If $b = -$ and $\Gamma = \Gamma', M$ then we need a natural isomorphism $[\![\Gamma]\!] \circ (\mathsf{prod} \times \mathsf{id}) \cong \mathsf{prod} \circ \mathsf{FamInj}([\![\Gamma]\!]) \circ \mathsf{dst}$. This is given by $[\![\Gamma]\!] \circ (\mathsf{prod} \times \mathsf{id}) = \oslash \circ ([\![\Gamma']\!] \times i[\![M]\!] \circ \pi_2) \circ \Delta \circ (\mathsf{prod} \times \mathsf{id}) = \oslash \circ ([\![\Gamma']\!] \circ (\mathsf{prod} \times \mathsf{id}) \times i[\![M]\!]) \circ (\mathsf{id} \times \Delta_{\mathcal{M}^{\Theta}_X}) \cong_{\mathsf{dist}_{-,\Gamma'}} \oslash \circ (\mathsf{prod} \circ \mathsf{FamInj}([\![\Gamma']\!]) \circ \mathsf{dst} \times i[\![M]\!]) \circ (\mathsf{id} \times \Delta) = \oslash \circ (\mathsf{prod} \times \mathsf{id}) \circ (\mathsf{FamInj}([\![\Gamma']\!]) \circ \mathsf{dst} \times i[\![M]\!]) \circ (\mathsf{id} \times \Delta) \cong_{\mathsf{dist}_{\oslash}} \mathsf{prod} \circ \mathsf{FamInj}(\oslash) \circ \mathsf{dst} \circ (\mathsf{FamInj}([\![\Gamma']\!]) \circ \mathsf{dst} \times i[\![M]\!]) \circ (\mathsf{id} \times \Delta) = \mathsf{prod} \circ \mathsf{FamInj}(\oslash) \circ \mathsf{FamInj}([\![\Gamma']\!] \times i[\![M]\!]) \circ \mathsf{dst} \circ (\mathsf{dst} \times \mathsf{id}) \circ (\mathsf{id} \times \Delta) = \mathsf{prod} \circ \mathsf{FamInj}(\oslash) \circ \mathsf{FamInj}([\![\Gamma']\!] \times i[\![M]\!] \circ (\mathsf{id} \times \Delta)) \circ \mathsf{dst} = \mathsf{prod} \circ \mathsf{FamInj}(\Gamma) \circ \mathsf{dst}$.
- If $b = -$ and $\Gamma = \Gamma', P$ then the transformation can be given similarly, using $\multimap$ instead of $\oslash$ and $\mathsf{dist}_{\multimap}$ instead of $\mathsf{dist}_{\oslash}$.

If $b = +$ the natural isomorphisms are given similarly, using $\oslash$ for positive formulas in the inductive step and $\multimap$ for negative formulas.

Finally, given a sequent A, Γ we define dist_{Γ} as the following horizontal composition, where b is the polarity of A. One can check pointwise that the functors are equal to the

given decompositions.

$$\begin{array}{ccccccc}
\forall x.[\![A,\Gamma]\!] : \mathcal{M}_X^\Theta & \xrightarrow{\langle \mathsf{add}_x, \mathsf{id}\rangle} & \mathsf{FamInj}(\mathcal{M}_{X\uplus\{x\}}^\Theta) \times \mathcal{M}_X^\Theta & \xrightarrow{\mathsf{FamInj}(A)\times \mathsf{id}} & \mathsf{FamInj}(\mathcal{G}_s) \times \mathcal{M}_X^\Theta & \xrightarrow{\mathsf{prod} \circ \mathsf{FamInj}([\![\Gamma]\!]_1^b) \circ \mathsf{dst}} & \mathcal{G}_s \\
\Big\Vert \mathsf{id} & & \Big\Vert \mathsf{id} & & \Big\Vert \mathsf{dist}_{b,\Gamma}^{-1} & & \\
[\![\forall x.A,\Gamma]\!] : \mathcal{M}_X^\Theta & \xrightarrow{\langle \mathsf{add}_x, \mathsf{id}\rangle} & \mathsf{FamInj}(\mathcal{M}_{X\uplus\{x\}}^\Theta) \times \mathcal{M}_X^\Theta & \xrightarrow{\mathsf{FamInj}(A)\times \mathsf{id}} & \mathsf{FamInj}(\mathcal{G}_s) \times \mathcal{M}_X^\Theta & \xrightarrow{[\![\Gamma]\!]_1^b \circ (\mathsf{prod}\times \mathsf{id})} & \mathcal{G}_s
\end{array}$$

Since dist_Γ is a natural isomorphism, and pointwise winning, it is an isomorphism in $\mathcal{W}^{\mathcal{M}_X^\Theta}$.

Proposition 4.2.8 $\pi_{v(x)} \circ \mathsf{dist}_{\Gamma(L,v)} = [\![\Gamma]\!]^b(\pi_{v(x)})$

Proof We can check this by induction on Γ, as in Proposition 2.3.5. ∎

4.2.5 Semantics of Proofs

We give semantics of a proof of $X;\Theta \vdash M,\Gamma$ as a uniform winning strategy $\kappa_I \Rightarrow [\![X;\Theta \vdash M,\Gamma]\!]$, and semantics of a proof of $X;\Theta \vdash P,\Gamma$ as a uniform winning strategy $[\![X;\Theta \vdash P,\Gamma]\!] \Rightarrow \kappa_\perp$.

We first introduce some notation. Suppose $\mathcal{C}$ is the coproduct of two categories $\mathcal{D}$ and $\mathcal{E}$ (the disjoint union of the two categories, where there are no maps between them). If $F : \mathcal{C} \to \mathcal{G}_e$ we write $F|_\mathcal{D}$ and $F|_\mathcal{E}$ for the restriction of F to $\mathcal{D}$ and $\mathcal{E}$ respectively. If $\eta : F \Rightarrow G$ then we can restrict η to a natural transformation $F|_\mathcal{D} \Rightarrow G|_\mathcal{D}$, and we write $\eta|_\mathcal{D}$ for this restriction. If $\eta : F|_\mathcal{D} \Rightarrow G|_\mathcal{D}$ and $\sigma : F|_\mathcal{E} \Rightarrow G|_\mathcal{E}$ then we write $[\eta,\sigma]_{\mathcal{D},\mathcal{E}}$ for the lax natural transformation defined by $[\eta,\sigma]_A = \eta_A$ if $A \in \mathcal{D}$ and $[\eta,\sigma]_A = \sigma_A$ if $A \in \mathcal{E}$. Lax naturality of $[\eta,\sigma]$ inherits from lax naturality of η and σ, since there are no maps between $\mathcal{D}$ and $\mathcal{E}$ when viewed as subcategories of $\mathcal{C}$.

We will sometimes abuse notation, writing $[\eta,\sigma]_{\mathcal{D},\mathcal{E}}$ even when there are maps between $\mathcal{D}$ and $\mathcal{E}$: we must then justify lax naturality of $[\eta,\sigma]_{\mathcal{D},\mathcal{E}}$ directly. If $\mathcal{C} = \mathcal{M}_X^\Theta$ then we will write $[\eta,\sigma]_{\alpha,\beta}$ for $[\eta,\sigma]_{\mathcal{M}_X^{\Theta,\alpha},\mathcal{M}_X^{\Theta,\beta}}$.

First, we deal with the semantics of the rules from WS!. We need an interpretation of each WS! rule: a transformation from the uniform winning strategies on each of the premises, to a uniform winning strategy on the conclusion. The value of $X;\Theta$ is constant throughout. By Proposition 4.2.5, $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a WS!-category. Thus we can use the interpretation of each WS! rule in this category. We next turn to the new rules.

Positive Atoms: For $\mathsf{P}_{\mathsf{at}+}$, we start with a lax natural transformation $[\![p]\!] : [\![\top,\Gamma]\!] \Rightarrow \perp$ with functors mapping $\mathcal{M}_X^{\Theta,\overline{\phi}(\overrightarrow{s})} \to \mathcal{G}_e$. But for any (L,v) in $\mathcal{M}_X^{\Theta,\overline{\phi}(\overrightarrow{s})}$ we have

$[\![\overline{\phi}(\overrightarrow{s}), \Gamma]\!](L, v) = [\![\top, \Gamma]\!](L, v)$. Hence $[\![p]\!] : [\![\overline{\phi}(\overrightarrow{s}), \Gamma]\!] \Rightarrow \bot$, and we take $[\![\mathsf{P}_{\mathsf{at}+}(p)]\!] = [\![p]\!]$.

Negative Atoms: For the rule $\mathsf{P}_{\mathsf{at}-}$, suppose $[\![p]\!] : I \Rightarrow [\![\bot, \Gamma]\!]$ with functors $\mathcal{M}_X^{\Theta, \overline{\phi}(\overrightarrow{s})} \to \mathcal{G}$. Then set $[\![\mathsf{P}_{\mathsf{at}-}(p)]\!](L, v) = \epsilon$ if $(L, v) \models \phi(\overrightarrow{s})$ and $[\![p]\!](L, v)$ if $(L, v) \models \overline{\phi}(\overrightarrow{s})$. In our above notation, this is $[[\![p]\!], \epsilon]_{\overline{\phi}(\overrightarrow{s}), \phi(\overrightarrow{s})}$.

For lax naturality, we need to check that the appropriate diagram lax commutes:

$$\begin{array}{ccc} I & \xrightarrow{[\![\mathsf{P}_{\mathsf{at}-}(p)]\!](M,w)} & [\![\phi(\overrightarrow{s}), \Gamma]\!](M, w) \\ {\scriptstyle \mathsf{id}}\downarrow & \sqsupseteq & \downarrow{\scriptstyle i[\![\phi(\overrightarrow{s}), \Gamma]\!](f)} \\ I & \xrightarrow[[\![\mathsf{P}_{\mathsf{at}-}(p)]\!](L,v)]{} & [\![\phi(\overrightarrow{s}), \Gamma]\!](L, v) \end{array}$$

If (L, v) and (M, w) agree on $\phi(\overrightarrow{x})$ then the diagram lax commutes by lax naturality of ϵ or $[\![p]\!]$. If they disagree, then we must have $(L, v) \models \overline{\phi}(\overrightarrow{x})$ and $(M, w) \models \phi(\overrightarrow{x})$. We need to show that $[\![\mathsf{P}_{\mathsf{at}-}(p)]\!](L, v) \sqsupseteq i[\![\phi(\overrightarrow{x}), \Gamma]\!](f) \circ [\![\mathsf{P}_{\mathsf{at}-}(p)]\!](M, w)$. To see this, note that $p[\![\phi(\overrightarrow{x}), \Gamma]\!](f) \circ [\![\mathsf{P}_{\mathsf{at}-}(p)]\!](L, v) = [\![\mathsf{P}_{\mathsf{at}-}(p)]\!](M, w)$ as both sides map into the terminal object, so $[\![\mathsf{P}_{\mathsf{at}-}(p)]\!](L, v) \sqsupseteq i[\![\phi(\overrightarrow{x}), \Gamma]\!](f) \circ p[\![\phi(\overrightarrow{x}), \Gamma]\!](f) \circ [\![\mathsf{P}_{\mathsf{at}-}(p)]\!](L, v) = i[\![\phi(\overrightarrow{x}), \Gamma]\!](f) \circ [\![\mathsf{P}_{\mathsf{at}-}(p)]\!](M, w)$.

Forall: Given a proof p of $X \uplus \{X\}; \Theta \vdash N, \Gamma$, then $[\![p]\!] : I \to [\![N, \Gamma]\!]$ in $\mathcal{W}^{\mathcal{M}_{X \uplus \{x\}}^{\Theta}}$. We can construct $\widehat{[\![p]\!]} : I \to \forall x.[\![N, \Gamma]\!]$ in $\mathcal{W}^{\mathcal{M}_X^{\Theta}}$ and set $[\![\mathsf{P}_{\forall}(p)]\!] = \mathsf{dist}_{\Gamma}^{-1} \circ \widehat{[\![p]\!]}$.

Exists: Consider the map $\mathsf{set}_s'^x(\eta) : \forall x.F = \mathsf{set}_s'^x(U_x'(\forall x.F)) \to \mathsf{set}_s'^x(F)$ in the category $\mathcal{M}_X^{\Theta}$. Pointwise, $\mathsf{set}_s'^x(\eta)_{(L,v)} : \prod_{l \in L} F(L, v[x \mapsto l]) \to F(L, v[x \mapsto v(s)])$ is given by $\pi_{v(s)}$, and so we will write π_s for this map.

Given a proof p of $X; \Theta \vdash P[s/x], \Gamma$ with $FV(s) \subseteq X$ we have a map $[\![p]\!] : [\![P[s/x], \Gamma]\!] \to \bot$ in $\mathcal{W}^{\mathcal{M}_X^{\Theta}}$. Since x does not occur in Γ, the domain of this map is $[\![(P, \Gamma)[s/x]]\!] = \mathsf{set}_s'^x([\![P, \Gamma]\!])$. Then we take $[\![\mathsf{P}_{\exists}^s(p)]\!] = [\![p]\!] \circ \pi_s \circ \mathsf{dist}_{\Gamma} = [\![p]\!] \circ [\![\Gamma]\!]^b(\pi_s) : [\![\forall x.P, \Gamma]\!] \to \bot$.

Inequality: Semantics of $\mathsf{P}_{\neq}$ is the empty family, as $\mathcal{M}_X^{\Theta}$ has no objects.

Match: For $\mathsf{P}_{\mathsf{ma}}^{x,y,z}$, we first construct an isomorphism

$$H_{x,y,z} : \mathcal{M}_X^{\Theta, x=y} \cong \mathcal{M}_{X/\{x,y\} \uplus \{z\}}^{\Theta[\frac{z}{x}, \frac{z}{y}]} : H_{x,y,z}^{-1}$$

with $H_{x,y,z}(M,v) = (M, v[z \mapsto v(x)] - x - y)$ and $H^{-1}(M,v) = (M, v[x \mapsto v(z), y \mapsto v(z)] - z)$. We can show that $[\![(X;\Theta \vdash \Gamma)[\frac{z}{x},\frac{z}{y}]]\!] = [\![X;\Theta, x = y \vdash \Gamma]\!] H^{-1}_{x,y,z}$ by induction on Γ.

We set $[\![\mathsf{P}^{x,y,z}_{\mathsf{ma}}(p,q)]\!] = [[\![p]\!]H_{x,y,z}, [\![q]\!]]_{x=y,x\neq y}$. The component of $[\![p^{x,y,z}_{\mathsf{ma}}(p,q)]\!]$ at (M,v) is given by $[\![p]\!]$ if $(M,v) \models x = y$ or $[\![q]\!]$ if $(M,v) \models x \neq y$.

Semantics of WS1 are given in Figure 4-3.

Figure 4-3: Semantics for WS1 — extends Figure 3-6

$$\mathsf{P}_{\mathsf{at}-}\ \frac{\sigma : [\![X;\Theta, \overline{\phi}(\overrightarrow{s}) \vdash \bot, \Gamma]\!]}{[\sigma,\epsilon]_{\overline{\phi}(\overrightarrow{s}),\phi(\overrightarrow{s})} : [\![X;\Theta \vdash \phi(\overrightarrow{s}), \Gamma]\!]}$$

$$\mathsf{P}_{\mathsf{at}+}\ \frac{\sigma : [\![X;\Theta, \overline{\phi}(\overrightarrow{s}) \vdash \top, \Gamma]\!]}{\sigma : [\![X;\Theta, \overline{\phi}(\overrightarrow{s}) \vdash \overline{\phi}(\overrightarrow{x}), \Gamma]\!]}$$

$$\mathsf{P}^{x,y,z}_{\mathsf{ma}}\ \frac{\sigma : [\![(X;\Theta \vdash \Gamma)[\frac{z}{x},\frac{z}{y}]]\!] \qquad \tau : [\![X;\Theta, x \neq y \vdash \Gamma]\!]}{[\sigma H_{x,y,z}, \tau]_{x=y,x\neq y} : [\![X;\Theta \vdash \Gamma]\!]}$$

$$\mathsf{P}_{\neq}\ \frac{}{\emptyset : [\![X;\Theta, x \neq x \vdash \Gamma]\!]}$$

$$\mathsf{P}_{\forall}\ \frac{\sigma : [\![X \uplus \{x\};\Theta \vdash N, \Gamma]\!]}{\mathsf{dist}^{-1}_{\Gamma} \circ \hat{\sigma} : [\![X;\Theta \vdash \forall x.N, \Gamma]\!]}\ x \notin FV(\Theta,\Gamma)$$

$$\mathsf{P}^{s}_{\exists}\ \frac{\sigma : [\![X;\Theta \vdash P[s/x], \Gamma]\!]}{\sigma \circ \pi_s \circ \mathsf{dist}_{\Gamma} : [\![X;\Theta \vdash \exists x.P, \Gamma]\!]}\ FV(s) \subseteq X$$

$$\mathsf{P}^{\mathsf{T}}_{\forall}\ \frac{\sigma : [\![X;\Theta \vdash M, \Gamma, \forall x.N, \Delta]\!]}{[\![\Delta]\!]^{-}(\mathsf{id} \oslash \pi_s) \circ \sigma : [\![X;\Theta \vdash M, \Gamma, N[s/x], \Delta]\!]}\ FV(s) \subseteq X$$

$$\mathsf{P}^{\mathsf{T}}_{\forall}\ \frac{\sigma : [\![X;\Theta \vdash Q, \Gamma, \forall x.N, \Delta]\!]}{\sigma \circ [\![\Delta]\!]^{+}(\pi_s \multimap \mathsf{id}) : [\![X;\Theta \vdash Q, \Gamma, N[s/x], \Delta]\!]}\ FV(s) \subseteq X$$

$$\mathsf{P}^{\mathsf{T}}_{\exists}\ \frac{\sigma : [\![X;\Theta \vdash M, \Gamma, P[s/x], \Delta]\!]}{[\![\Delta]\!]^{-}(\pi_s \multimap \mathsf{id}) \circ \sigma : [\![X;\Theta \vdash M, \Gamma, \exists x.P, \Delta]\!]}\ FV(s) \subseteq X$$

$$\mathsf{P}^{\mathsf{T}}_{\exists}\ \frac{\sigma : [\![X;\Theta \vdash Q, \Gamma, P[s/x], \Delta]\!]}{\sigma \circ [\![\Delta]\!]^{+}(\mathsf{id} \oslash \pi_s) : [\![X;\Theta \vdash Q, \Gamma, \exists x.P, \Delta]\!]}\ FV(s) \subseteq X$$

$$\mathsf{P}_{\mathsf{fneq}}\ \frac{\sigma : [\![X;\Theta, s \neq t \vdash \Gamma]\!]}{\sigma|_{s\neq t} : [\![X;\Theta, f(s) \neq f(t) \vdash \Gamma]\!]}$$

4.3 Full Completeness

We next show a full completeness result for the *function-free* fragment of $\mathsf{WS1}$: we henceforth assume that $\mathcal{L}$ contains no function symbols. Thus, the only uses of the $\mathsf{P}_\exists$ rule are of the form $\mathsf{P}_\exists^y$ where y is some variable in scope.

We show that the core rules suffice to represent any uniform winning strategy σ on a type object provided σ is *bounded* — i.e. there is a bound on the size of plays occurring in σ. In particular, such a strategy is the semantics of a unique analytic proof. If σ is not bounded, then it is the semantics of a unique infinitary analytic proof. We extend our reification procedure for $\mathsf{WS!}$ to $\mathsf{WS1}$.

We assume some arbitrary linear ordering on $\mathcal{V}$, which lifts lexicographically to $\mathcal{V} \times \mathcal{V}$. Let $\mathsf{Fr}(X; \Theta \vdash \Gamma)$ denote $\mathcal{V} - (X \cup B(\Gamma))$ where $B(\Gamma)$ denotes the variables that are bound in Γ.

Definition Given a sequent $X; \Theta \vdash \Gamma$, Θ is *lean* if:

- Θ contains $x \neq y$ for all distinct x and y in X
- Θ does not contain $x \neq x$ for any variable x.

A proof in $\mathsf{WS1}$ is *analytic* if it uses only core rules and has the following additional restrictions:

- Rules other than $\mathsf{P}_\neq$ and $\mathsf{P}_{\mathsf{ma}}^{x,y,z}$ can only conclude sequents with a lean Θ
- If $\mathsf{P}_{\mathsf{ma}}^{x,y,z}$ is used to conclude $X; \Theta \vdash \Gamma$ then X does not contain $w \neq w$ for any w; (x, y) is the least pair with $x, y \in X$, $x \not\equiv y$ and $x \neq y \notin \Theta$; and z is the least variable in $\mathsf{Fr}(X; \Theta \vdash \Gamma)$ (the least fresh variable).

We will show:

Theorem 4.3.1 *Let $X; \Theta \vdash \Gamma$ be a sequent of WS1. If σ is a bounded uniform winning strategy on $[\![X; \Theta \vdash \Gamma]\!]$ then there is a unique analytic proof p of $X; \Theta \vdash \Gamma$ with $[\![p]\!] = \sigma$.*

We will also extend this result to reification of unbounded strategies as infinitary analytic proofs, as in Chapter 3. We can hence normalise proofs to their (possibly infinitary) analytic form.

4.3.1 Uniform Choice

First, we show that for any uniform winning strategy, each component makes the same choice (in some sense) when the outermost connective is $\oplus$ or $\exists$.

Proposition 4.3.2 *If Θ is lean and $(L,v),(M,w) \in \mathcal{M}_X^\Theta$ there exists an $\mathcal{L}$-model $(L,v)\sqcup(M,w)$ with maps $f_{(L,v,M,w)} : (L,v) \to (L,v)\sqcup(M,w)$ and $g_{(L,v,M,w)} : (M,w) \to (L,v) \sqcup (M,w)$.*

Proof If (L,v) is an $\mathcal{L}$-model, define $U_{(L,v)}$ to be the elements of $|L|$ not in the image of v. Then the carrier of $(L,v) \sqcup (M,w)$ is defined to be $X \uplus U_{(L,v)} \uplus U_{(M,w)}$. The $\mathcal{L}$-structure validates all positive atoms, and the valuation is just inj_1. Then the map $f_{(L,v,M,w)}$ sends $v(x)$ to $\mathsf{inj}_1(x)$ and $u \in U_{(L,v)}$ to $\mathsf{inj}_2(u)$. This is an injection because Θ is lean. $g_{(L,v,M,w)}$ is defined similarly. ∎

We also recall that if $f : (L,v) \to (M,w)$ then $\sigma_{(L,v)}$ is determined entirely by f and $\sigma_{(M,w)}$. In particular, uniformity for positive strategies $\sigma : N \Rightarrow \bot$ requires that $\sigma_{(L,v)} \sqsubseteq \sigma_{(M,w)} \circ N(f)$ but since $\sigma_{(L,v)}$ is total, it is maximal in the ordering and so we must have $\sigma_{(L,v)} = \sigma_{(M,w)} \circ N(f)$.

Proposition 4.3.3 *Let $X;\Theta \vdash \Gamma$ be a sequent and suppose Θ is lean. Then there exists an object in $\mathcal{M}_X^\Theta$.*

Proof Note that Θ just contains positive atoms. We can take (X,id), with $(X,\mathsf{id}) \models \overline{\phi}(\overrightarrow{x})$ just if $\overline{\phi}(\overrightarrow{x}) \in \Theta$. Then each formula in Θ is satisfied: each such formula is either $\overline{\phi}(\overrightarrow{x})$, or $x \neq y$ for distinct x, y. ∎

Proposition 4.3.4 *Let $M_1, M_2 : \mathcal{M}_X^\Theta \to \mathcal{G}_e$. Suppose Θ is lean, and let $\sigma : M_1 \times M_2 \Rightarrow \bot$ be a uniform total (resp. winning) strategy. Then $\sigma = \tau \circ \pi_1$ for some uniform total (resp. winning) strategy $\tau : M_1 \Rightarrow \bot$, or $\sigma = \tau \circ \pi_2$ for some uniform total (resp. winning) strategy $\tau : M_2 \Rightarrow \bot$.*

Proof We know that each $\sigma_{(L,v)}$ is of the form $\tau_{(L,v)} \circ \pi_i$ for some $i \in \{1,2\}$ since in the game $M_1(L,v) \times M_2(L,v) \multimap \bot$ we must respond to the initial Opponent-move either with a move in M_1 or a move in M_2 (the π-atomicity condition). But we need to check that i is uniform across components. Suppose that i is not uniform — then we have (L,v) and (T,w) with $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_1$ and $\sigma_{(T,w)} = \tau_{(T,w)} \circ \pi_2$. Now consider $(L,v) \sqcup (T,w)$ and let k be such that $\sigma_{(L,v)\sqcup(T,w)} = \tau_{(L,v)\sqcup(T,w)} \circ \pi_k$. By uniformity and totality, $\sigma_{(L,v)} = \sigma_{(L,v)\sqcup(T,w)} \circ (M_1 \times M_2)(f_{(L,v,T,w)}) = \tau_{(L,v)\sqcup(T,w)} \circ \pi_k \circ (M_1 \times M_2)(f_{(L,v,T,w)}) = \tau_{(L,v)\sqcup(T,w)} \circ M_k(f_{(L,v,T,w)}) \circ \pi_k$. But since $\sigma_{(L,v)}$ is of the form $\tau_{(L,v)} \circ \pi_1$, we must have $k = 1$. But we can reason similarly using $\sigma_{(T,w)}$ and $g_{(L,v,T,w)}$ and discover that $k = 2$. This is a contradiction.

Thus there is some i such that each $\sigma_{(L,v)}$ can be decomposed into $\tau_{(L,v)} \circ \pi_i$. In particular, we can take i such that $\sigma_{(X,\mathsf{id})} = \tau_{(X,\mathsf{id})} \circ \pi_i$ where (X,id) is as defined in

Proposition 4.3.3. We only need to show that τ is lax natural. We can construct a natural transformations $\iota_1 : \langle \mathsf{id}, \epsilon \rangle : M_1 \to M_1 \times M_2$ and $\iota_2 : \langle \epsilon, \mathsf{id} \rangle : M_2 \to M_1 \times M_2$. Then $\tau = \sigma \circ \iota_i$, and so is lax natural. ∎

Corollary 4.3.5 *If Θ is lean, then $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a complete WS-category.*

Proof We show that each of the conditions of complete WS-categories hold.

1a There are no maps $I \to \bot$. If there were, we could take its component at (X, id) yielding a winning strategy on $I \multimap \bot$, of which there are none.

1b Proposition 4.3.4 ensures that the π-atomicity axiom holds.

2 The map $(\_ \multimap \bot)^{-1}$ on $\mathcal{W}$ extends pointwise to $\mathcal{W}^{\mathcal{M}_X^\Theta}$. ∎

Proposition 4.3.6 *Let $M : \mathcal{M}^\Theta_{X \uplus \{x\}} \to \mathcal{G}_e$. Suppose Θ is lean, and let $\sigma : \forall x.M \Rightarrow \bot$ be a uniform total (resp. winning) strategy. Then there exists a unique variable $y \in X$ and uniform total (resp. winning) strategy $\tau : M\mathsf{set}^x_y \Rightarrow \bot$ such that $\sigma = \tau \circ \pi_y$.*

Proof We firstly show that given any $\mathcal{L}$-model (L, v) there is some x with $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_{v(x)}$. Suppose for contradiction that $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_u$ for some $u \in U_{(L,v)}$. Build the $\mathcal{L}$-model $L' = X \uplus \{a, b\} \uplus U_{(L,v)}$ with valuation inj_1 and validating all positive atoms. Let $\sigma_{(L', \mathsf{inj}_1)} = \tau_{(L', \mathsf{inj}_1)} \circ \pi_r$. Define $m_1 : (L, v) \to (L', \mathsf{inj}_1)$ sending $v(x)$ to $\mathsf{inj}_1(x)$, u to $\mathsf{inj}_2(a)$ and $y \in U_{(L,v)} - \{u\}$ to $\mathsf{inj}_3(y)$. Then $\sigma_{(L,v)} = \sigma_{(L', \mathsf{inj}_1)} \circ \pi_r \circ \forall x.M(m_1)$.

- If $r = \mathsf{inj}_1(x)$ then this is $\sigma_{(L', \mathsf{inj}_1)} \circ M(m_1) \circ \pi_{v(x)}$, contradicting $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_u$.
- If $r = \mathsf{inj}_2(b)$ then this is $\sigma_{(L', \mathsf{inj}_1)} \circ \epsilon$ which is ϵ as $\sigma_{(L', \mathsf{inj}_1)}$ is strict (it is a total map into $\bot$). This is impossible as $\sigma_{(L,v)}$ is total.
- If $r = \mathsf{inj}_3(y)$ then this is $\sigma_{(L', \mathsf{inj}_1)} \circ M(m_1) \circ \pi_y$, , contradicting $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_u$.
- Hence we must have $r = \mathsf{inj}_2(a)$.

Define $m_2 : (L, v) \to (L', \mathsf{inj}_1)$ sending $v(x)$ to $\mathsf{inj}_1(x)$, u to $\mathsf{inj}_2(b)$ and $y \in U_{(L,v)} - \{u\}$ to $\mathsf{inj}_3(y)$. We can use similar reasoning to show that $r = \mathsf{inj}_2(b)$. This is a contradiction.

Hence, given any (L, v) there is some variable x such that $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_{v(x)}$. Let $y \in X$ be the unique variable such that $\sigma_{(X,\mathsf{id})} = \tau_{(X,\mathsf{id})} \circ \pi_y$ where (X, id) is constructed as in Proposition 4.3.3. We now show that for all (L, v), $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_{v(y)}$. Suppose that $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_{v(x)}$ and $\sigma_{(L,v) \sqcup (X,\mathsf{id})} = \tau_{(L,v) \sqcup (X,\mathsf{id})} \circ \pi_{\mathsf{inj}_1(z)}$. By lax naturality, $\tau_{(L,v)} \circ \pi_{v(x)} = \sigma_{(L,v)} = \sigma_{(L,v) \sqcup (X,\mathsf{id})} \circ \forall x.M(f_{(L,v,X,\mathsf{id})}) = \tau_{(L,v) \sqcup (X,\mathsf{id})} \circ \pi_{\mathsf{inj}_1(z)} \circ \forall x.M(f_{(L,v,X,\mathsf{id})})$. Since $\mathsf{inj}_1(z) = f_{(L,v,X,\mathsf{id})}(v(z))$, we have $\sigma_{(L,v)} = \tau_{(L,v) \sqcup (X,\mathsf{id})} \circ M(f_{(L,v,X,\mathsf{id})}) \circ \pi_{v(z)}$ and so we must have $x = z$. By similar reasoning using $g_{(L,v,X,\mathsf{id})}$, we see that $y = z$, so $x = y$.

Hence there is a variable y such that for all (L, v), $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_{v(y)}$ for some $\tau_{(L,v)} : M(L, v[x \mapsto v(y)]) \Rightarrow \bot$. Since Θ is lean, y is the unique variable such that $\sigma_{(L,v)} = \tau_{(L,v)} \circ \pi_{v(y)}$. Note that $M(L, v[x \mapsto v(y)]) = M(\mathsf{set}^x_y(L, v))$. It only remains to show that the resulting transformation $\tau : M\mathsf{set}^x_y \Rightarrow \bot$ is lax natural.

Consider the transformation $\rho : M\mathsf{set}^x_y \Rightarrow \forall x.M : \mathcal{M}^\Theta_X \to \mathcal{G}_s$ defined by $\rho(L, v) = \langle g_m \rangle_m$ where $g_m = \epsilon$ if $m \neq v(y)$ and $g_{v(y)} = \mathsf{id}$. We show that this is natural:

$$\begin{array}{ccc} M(L,v) & \xrightarrow{\rho_{L,v}} & \forall x.M(L,v) \\ \Big\downarrow{\scriptstyle M(f)} & & \Big\downarrow{\scriptstyle \forall x.M(f)} \\ M(M,w) & \xrightarrow[\rho_{M,w}]{} & \forall x.M(M,w) \end{array}$$

We show $\pi_b \circ \rho_{M,w} \circ M(f) = \pi_b \circ \forall x.M(f) \circ \rho_{L,v}$ for each b.

- If $b \neq w(y)$ and is not in the image of f, then the LHS is $\epsilon \circ M(f) = \epsilon$ and the RHS is $\epsilon \circ \rho_{L,v} = \epsilon$ as required.
- If $b = f(a) \neq w(y)$ then the LHS is still ϵ, and the RHS is $M(f) \circ \pi_a \circ \rho_{L,v}$. But $a \neq v(y)$ (since $a = v(y) \Rightarrow b = f(a) = w(y)$) and so this is $M(f) \circ \epsilon$. Since $M(f)$ is strict, this is ϵ, as required.
- If $b = w(y)$ then the LHS is $\mathsf{id} \circ M(f) = M(f)$ and the RHS is $M(f) \circ \pi_{v(y)} \circ \rho_A = M(f) \circ \mathsf{id} = M(f)$ as required.

Finally, we can see that τ is lax natural because $\tau = \sigma \circ \rho$. ∎

4.3.2 Reification Procedure

We next define a function reify from bounded uniform winning strategies on $[\![X; \Theta \vdash \Gamma]\!]$ to proofs of $X; \Theta \vdash \Gamma$. This extends the procedure given in previous chapters. Again it is defined by case analysis on the head of Γ, by induction on a compound measure involving the size of the strategy, the number of pairs of free variables that are not declared distinct by Θ, and a further measure which depends on the nature of the head formula. Informally, if Θ is not lean:

- If Θ contains $x \neq x$ we use $\mathsf{P}_{\neq}$ and halt.
- Otherwise, we consider the least two variables $x, y \in X$ that are not declared distinct by Θ and split the family into those models that identify x and y, and those that do not. In the former case, we can substitute fresh z for both x and y. We then apply the inductive hypothesis to both halves and apply $\mathsf{P}^{x,y,z}_{\mathsf{ma}}$ using $H^{-1}_{x,y,z}$.

If Θ is lean, then:

- If the head formula is not an atom or quantifiers, we proceed as with reification as in Figure 3-7, using the fact that $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a complete WS-category.
- If the head formula is a positive atom $\overline{\phi}(\overrightarrow{x})$ then we must have $\overline{\phi}(\overrightarrow{x})$ in Θ, as otherwise there can be no uniform winning strategies on $[\![\Gamma]\!]$ (since some games in that family have no winning strategies). Thus we can proceed inductively and apply $\mathsf{P}_{\mathsf{at}+}$.
- If the head formula is a negative atom $\phi(\overrightarrow{x})$ then we can split the family σ into those models that satisfy $\phi(\overrightarrow{x})$ and those that do not. All strategies in the former set must be empty, as there are no moves to play. All strategies in the latter set form a uniform winning strategy on $[\![\Theta, \overline{\phi}(\overrightarrow{x}) \vdash \bot, \Gamma]\!]$ and we can proceed inductively using $\mathsf{P}_{\mathsf{at}-}$.
- If $\sigma : [\![X; \Theta \vdash \Gamma = \forall x.N, \Gamma']\!]$ then $\mathsf{dist}_{\Gamma'} \circ \sigma : I \Rightarrow \forall x.[\![N, \Gamma']\!]$. Using our adjunction, this corresponds to a map $\eta \circ U'_x(\mathsf{dist}_{\Gamma'} \circ \sigma) : I \Rightarrow [\![N, \Gamma']\!]$ in $\mathcal{W}^{\mathcal{M}^\Theta_{X \uplus \{x\}}}$. We can then reify this inductively to yield a proof of $X \uplus \{x\}; \Theta \vdash N, \Gamma'$ and apply $\mathsf{P}_\forall$.
- If $\Gamma = \exists x.P, \Gamma'$ then $\sigma \circ \mathsf{dist}_{+,\Gamma'} : \forall x.[\![P, \Gamma']\!] \Rightarrow \bot$. By Proposition 4.3.6, there is a unique y and natural transformation $\tau : [\![P, \Gamma']\!]\mathsf{set}^x_y \Rightarrow \bot$ such that $\sigma \circ \mathsf{dist}_{+,\Gamma'} = \tau \circ \pi_y$. Since x does not occur in Γ, we have $[\![P, \Gamma']\!]\mathsf{set}^x_y = [\![P[y/x], \Gamma']\!]$. This yields a lax natural transformation $[\![P[y/x], \Gamma']\!] \Rightarrow \bot$. We can then apply the inductive hypothesis use the $\mathsf{P}^y_\exists$ rule.

We formally define reify for the new connectives in Figure 4-4. The other cases (with lean Θ) are imported from Figure 3-7, using the fact that $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a complete WS!-category.

Remark Note that if $\sigma : [\![\{x, y\}; \vdash \Gamma]\!]$ then the reification procedure places both $x \neq y$ and $y \neq x$ into the context, as either could (in principle) be required later in the proof. Assuming z is least in $\mathsf{Fr}(X; \Theta \vdash \Gamma)$ and $x \leqslant y$ in the ordering, $\mathsf{reify}(\sigma)$ is the following proof:

$$\mathsf{P}^{x,y,z}_{\mathsf{ma}} \dfrac{\dfrac{\mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta, x=y}} \circ H^{-1}_{x,y,z})}{\{z\}; \vdash \Gamma[\frac{z}{x}, \frac{z}{y}]} \quad \mathsf{P}^{y,x,z}_{\mathsf{ma}} \dfrac{\mathsf{P}_{\neq} \dfrac{}{\{z\}; z \neq z \vdash \Gamma[\frac{z}{x}, \frac{z}{y}]} \quad \dfrac{\mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta, x \neq y}})}{\{x, y\}; x \neq y, y \neq x \vdash \Gamma}}{\{x, y\}; x \neq y \vdash \Gamma}}{\{x, y\}; \vdash \Gamma}$$

Figure 4-4: Reification of Strategies — extends Figure 3-7

For non-lean Θ:

$$\begin{array}{lcl}
\mathsf{reify}_{X,x\neq x;\Theta\vdash\Gamma}(\sigma) & = & \mathsf{P}_{\not\equiv} \\
\mathsf{reify}_{X,x,y;\Theta\vdash\Gamma}(\sigma) & = & \mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta,x=y}} \circ H_{x,y,z}^{-1}), \mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta,x\neq y}})) \\
 & & \text{if } (x,y) \in X \times X \text{ is least such that } x \not\equiv y \text{ and } (x \neq y) \notin \Theta \\
 & & \text{and } z \text{ is the least element in } \mathsf{Fr}(X; \Theta \vdash \Gamma)
\end{array}$$

For lean Θ:

$$\begin{array}{lcl}
\mathsf{reify}_{X;\Theta\vdash\phi(\vec{x}),\Gamma}(\sigma) & = & \mathsf{P}_{\mathsf{at}-}(\mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta,\overline{\phi}(\vec{x})}})) \\
\mathsf{reify}_{X;\Theta\vdash\overline{\phi}(\vec{x}),\Gamma}(\sigma) & = & \mathsf{P}_{\mathsf{at}+}(\mathsf{reify}(\sigma)) \\
\mathsf{reify}_{X;\Theta\vdash\forall x.N,\Gamma}(\sigma) & = & \mathsf{P}_{\forall}(\mathsf{reify}(\eta \circ U'_x(\mathsf{dist}_{\Gamma'} \circ \sigma)) \\
\mathsf{reify}_{X;\Theta\vdash\exists x.N,\Gamma}(\sigma) & = & \mathsf{P}_{\exists}^{y}(\mathsf{reify}(\tau)) \text{ where } \sigma \circ \mathsf{dist}_{\Gamma}^{-1} = \tau \circ \pi_y
\end{array}$$

4.3.3 Termination

We next need to show that if σ is bounded then $\mathsf{reify}(\sigma)$ terminates. We can base our measure on that given in Proposition 2.4.3, but we need to add an extra component to our lexicographical measure to take into account the use of P_{ma} to reduce the number of distinct variables in X that are not declared distinct by Θ.

Again, the full completeness procedure first breaks down the head formula until it is $\bot$ or $\top$. It then uses the core elimination rules to compose the tail into (at most) a single formula. These steps do not increase the size of the strategy. Finally, the head is removed using $\mathsf{P}_{\bot}^{+}$ or $\mathsf{P}_{\top}^{-}$, strictly reducing the size of the strategy. If Θ is not lean, the number of distinct variable pairs that are not declared distinct in Θ is reduced by using P_{ma}.

Formally, we can see this as a lexicographical ordering of four measures on σ,X,Θ,Γ:

- The most dominant measure is the length of the longest play in σ.
- The second measure is the length of Γ as a list if the head of Γ is $\bot$ or $\top$, and ∞ otherwise.
- The third measure is the size of the head formula of Γ.
- The fourth measure is

$$\mathsf{L}(X,\Theta) = |\{(x,y) \in X \times X : x \not\equiv y \wedge x \neq y \notin \Theta\}|.$$

If Θ is lean:

- If $\Gamma = \bot, P$ or $\top, N$ then the first measure decreases in the call to the inductive hypothesis.

- Otherwise, if $\Gamma = A, \Gamma'$ with $A \in \bot, \top$ the first measure does not increase and the second measure decreases.
- If $\Gamma = A, \Gamma'$ with $A \notin \{\bot, \top\}$, the first measure does not increase and either the second or third measure decreases.

If Θ is not lean and the $\mathsf{P_{ma}}$ rule is applied, in the call to the inductive hypotheses the first three measures stay the same and the fourth measure decreases.

Thus, the inductive hypothesis is used with a smaller value in the compound measure on $\mathbb{N} \times \mathbb{N} \cup \{\infty\} \times \mathbb{N} \times \mathbb{N}$ ordered lexicographically.

4.3.4 Soundness and Uniqueness

Proposition 4.3.7 *Given any bounded uniform winning strategy* $\sigma : [\![X; \Theta \vdash \Gamma]\!]$, $[\![\mathsf{reify}(\sigma)]\!] = \sigma$.

Proof We proceed by induction on our termination measure. For the constructs that appear in WS!, we can proceed as in Proposition 3.6.1, since $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a complete WS!-category. For the new cases:

- If Θ is not lean and $x \neq x \in \Theta$ then we must have $[\![\mathsf{reify}(\sigma)]\!] = \sigma$ as there is a unique uniform winning strategy on this functor (the empty family).
- If Θ is not lean with $(x, y) \in X \times X$ least such that $x \not\equiv y$ and $(x \neq y) \notin \Theta$ and z is the least element in $\mathsf{Fr}(X; \Theta \vdash \Gamma)$, then $[\![\mathsf{reify}(\sigma)]\!]$
  $= [\![\mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta, x=y}} \circ H_{x,y,z}^{-1}, \mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta, x\neq y}})]\!]$
  $= [[\![\mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta, x=y}} \circ H_{x,y,z}^{-1})]\!] H_{x,y,z}, [\![\mathsf{reify}(\sigma|_{\mathcal{M}_X^{\Theta, x\neq y}})]\!]]_{x=y, x\neq y}$
  $= [\sigma|_{\mathcal{M}_X^{\Theta, x=y}} \circ H_{x,y,z}^{-1} \circ H_{x,y,z}, \sigma|_{\mathcal{M}_X^{\Theta, x\neq y}}]_{x=y, x\neq y}$
  $= [\sigma|_{\mathcal{M}_X^{\Theta, x=y}}, \sigma|_{\mathcal{M}_X^{\Theta, x\neq y}}]_{x=y, x\neq y} = \sigma$.
- If $\Gamma = \overline{\phi}(\overrightarrow{x}), \Gamma'$ then $[\![\mathsf{reify}(\sigma)]\!] = [\![\mathsf{P}_{\mathsf{at}+}(\mathsf{reify}_{\top, \Gamma'}(\sigma))]\!] = [\![\mathsf{reify}_{\top, \Gamma'}(\sigma)]\!] = \sigma$.
- If $\Gamma = \phi(\overrightarrow{x}), \Gamma'$ then $[\![\mathsf{reify}(\sigma)]\!] = [\![\mathsf{P}_{\mathsf{at}-}(\sigma|_{\mathcal{M}_X^{\Theta, \overline{\phi}(\overrightarrow{x})}})]\!] = [\sigma|_{\mathcal{M}_X^{\Theta, \overline{\phi}(\overrightarrow{x})}}, \epsilon]_{\overline{\phi}(\overrightarrow{x}), \phi(\overrightarrow{x})} = \sigma$ as we must have $\sigma|_{\mathcal{M}_X^{\Theta, \phi(\overrightarrow{x})}} = \epsilon$ since $[\![\phi(\overrightarrow{x}), \Gamma]\!]_A$ is the terminal object for each A in $\mathcal{M}_X^{\Theta, \phi(\overrightarrow{x})}$.
- If $\Gamma = \forall x.N, \Gamma'$ then $[\![\mathsf{reify}(\sigma)]\!] = [\![\mathsf{P}_\forall(\mathsf{reify}(\eta \circ U'_x(\mathsf{dist}_{\Gamma'} \circ \sigma)))]\!] =$
  $\mathsf{dist}_{\Gamma'}^{-1} \circ [\![\mathsf{reify}(\eta \circ \widehat{U'_x(\mathsf{dist}_{\Gamma'} \circ \sigma)})]\!] = \mathsf{dist}_{\Gamma'}^{-1} \circ (\eta \circ \widehat{U'_x(\mathsf{dist}_{\Gamma'}} \circ \sigma)) = \mathsf{dist}_{\Gamma'}^{-1} \circ \mathsf{dist}_{\Gamma'} \circ \sigma =$
  σ as required.
- If $\Gamma = \exists x.P, \Gamma'$ then $[\![\mathsf{reify}(\sigma)]\!] = [\![\mathsf{P}_\exists^y(\mathsf{reify}(\tau))]\!]$ where $\sigma \circ \mathsf{dist}_{\Gamma'}^{-1} = \tau \circ \pi_y$. This is $[\![\mathsf{reify}(\tau)]\!] \circ \pi_y \circ \mathsf{dist}_{\Gamma'} = \tau \circ \pi_y \circ \mathsf{dist}_{\Gamma'} = \sigma \circ \mathsf{dist}_{\Gamma'}^{-1} \circ \mathsf{dist}_{\Gamma'} = \sigma$ as required. ∎

Proposition 4.3.8 *For any analytic proof* p, $\mathsf{reify}([\![p]\!]) = p$.

Proof Induction on p. If the final rule used in p was a rule of WS! (used necessarily with lean Θ) we can proceed as in Proposition 3.6.2, noting that $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a complete WS!-category. Thus, we concentrate on the new rules.

- If $p = \mathsf{P}_{\neq}$ then we have $x \neq x$ in Θ, and so $\mathsf{reify}_{X,\Theta\vdash\Gamma}(\llbracket p \rrbracket) = \mathsf{P}_{\neq}$ as required.
- If $p = \mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1, p_2)$ with $(x, y) \in X \times X$ least such that $x \not\equiv y$ and $(x \neq y) \notin \Theta$ and z is the least element in $\mathsf{Fr}(X; \Theta \vdash \Gamma)$, then $\mathsf{reify}(\llbracket p \rrbracket) = \mathsf{reify}([\llbracket p_1 \rrbracket H_{x,y,z}, \llbracket p_2 \rrbracket]_{x=y, x\neq y}) = \mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{reify}([\llbracket p_1 \rrbracket H_{x,y,z}, \llbracket p_2 \rrbracket]_{x=y,x\neq y}|_{x=y} \circ H_{x,y,z}^{-1}), \mathsf{reify}([\llbracket p_1 \rrbracket H_{x,y,z}, \llbracket p_2 \rrbracket]_{x=y,x\neq y}|_{x\neq y})) = \mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{reify}(\llbracket p_1 \rrbracket), \mathsf{reify}(\llbracket p_2 \rrbracket)) = \mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1, p_2) = p$ as required.
- If $p = \mathsf{P}_{\mathsf{at}+}(q)$ then $\mathsf{reify}(\llbracket p \rrbracket) = \mathsf{P}_{\mathsf{at}+}(\mathsf{reify}(\llbracket q \rrbracket)) = \mathsf{P}_{\mathsf{at}+}(q) = p$ as required.
- If $p = \mathsf{P}_{\mathsf{at}-}(q)$ then $\mathsf{reify}(\llbracket p \rrbracket) = \mathsf{reify}([\llbracket q \rrbracket, \epsilon]_{\overline{\phi}(\vec{x}),\phi(\vec{x})}) = \mathsf{P}_{\mathsf{at}-}(\mathsf{reify}([\llbracket q \rrbracket, \epsilon]_{\overline{\phi}(\vec{x}),\phi(\vec{x})}|_{\overline{\phi}(\vec{x})})) = \mathsf{P}_{\mathsf{at}-}(\mathsf{reify}(\llbracket q \rrbracket)) = \mathsf{P}_{\mathsf{at}-}(q) = p$ as required.
- If $p = \mathsf{P}_{\exists}^{y}(q)$ then $\mathsf{reify}(\llbracket p \rrbracket) = \mathsf{reify}(\llbracket q \rrbracket \circ \pi_y \circ \mathsf{dist}_{\Gamma'})$. Since $\llbracket q \rrbracket \circ \pi_y \circ \mathsf{dist}_{\Gamma'} \circ \mathsf{dist}_{\Gamma'} = \llbracket q \rrbracket \circ \pi_y$, this is $\mathsf{P}_{\exists}^{y}(\mathsf{reify}(\llbracket q \rrbracket)) = \mathsf{P}_{\exists}^{y}(q) = p$ as required.
- If $p = \mathsf{P}_{\forall}(q)$ then $\mathsf{reify}(\llbracket p \rrbracket) = \mathsf{reify}(\mathsf{dist}_{\Gamma'}^{-1} \circ \hat{\llbracket q \rrbracket}) = \mathsf{P}_{\forall}(\mathsf{reify}(\eta \circ U'_x(\mathsf{dist}_{\Gamma'} \circ \mathsf{dist}_{\Gamma'}^{-1} \circ \hat{\llbracket q \rrbracket}))) = \mathsf{P}_{\forall}(\mathsf{reify}(\eta \circ U'_x(\hat{\llbracket q \rrbracket}))) = \mathsf{P}_{\forall}(\mathsf{reify}(\llbracket q \rrbracket)) = \mathsf{P}_{\forall}(q) = p$ as required. ∎

We thus see that for any bounded σ, $\mathsf{reify}(\sigma)$ is the unique analytic proof p such that $\llbracket p \rrbracket = \sigma$.

4.4 Proof Normalisation

We can extend our full completeness procedure to unbounded uniform total strategies, yielding infinitary analytic proofs. We can then normalise proofs in WS1 to their infinitary analytic forms. The treatment follows that in Section 3.7, and we describe how this approach can be extended to deal with WS1.

- The definition of infinitary analytic proof of WS1 follows the definitions in Section 3.7. In particular, we can formulate $\mathcal{I}_\Gamma$ using a final coalgebra (note that the set Seq now includes the $X; \Theta$ component). Alternatively, we can consider the set of infinitary analytic proofs as the limit of the analytic paraproofs, which are analytic proofs in WS1 with access to a diamon rule P_ϵ that can prove any sequent.
- We can give semantics of an analytic paraproof of $X; \Theta \vdash \Gamma$ as a uniform strategy on $\llbracket X; \Theta \vdash \Gamma \rrbracket$. For the rules of WS!, we use the fact that $\mathcal{G}^{\mathcal{M}_X^\Theta}$ is a WS!-category. We can interpret the new rules $\mathsf{P}_\forall$, $\mathsf{P}_\exists^y$, P_{ma}, $\mathsf{P}_{\neq}$, $\mathsf{P}_{\mathsf{at}+}$, $\mathsf{P}_{\mathsf{at}-}$ as given in Figure 4-3: the lack of winningness causes no problems.
- We can extend these semantics to infinitary analytic proofs of WS1.

- To do this, note that $\mathcal{G}^{\mathcal{M}_X^\Theta}$ is cpo-enriched, inheriting pointwise from $\mathcal{G}$. As before, the semantic operation $[\![-]\!]$ mapping $\mathcal{C}_{X;\Theta\vdash\Gamma}$ to the set of uniform strategies on $[\![X;\Theta\vdash\Gamma]\!]$ is monotonic. We can thus define the semantics of an infinitary analytic proof as the limit of its finite approximants.
- We can show that these semantics agree with those given in Figure 4-3 because the underlying semantic operations are continuous. First, composition, currying and pairing in $\mathcal{G}^{\mathcal{M}_X^\Theta}$ are continuous. Then we note that if σ and τ are continuous so is $[\sigma,\tau]$ and σH for any functor H. Finally, we can show that the $\hat{-}$ operation on uniform strategies is continuous by considering the pointwise definition and noting that arbitrary tupling is continuous.
- We can show that the resulting uniform strategy is total by showing that, for each A and n, σ_A is n-total. This is by lexicographic induction on

$$\langle n, \mathsf{tl}^+(\Gamma), \mathsf{hd}^+(\Gamma), \mathsf{tl}^-(\Gamma), \mathsf{hd}^-(\Gamma), \mathsf{L}(X,\Theta)\rangle$$

  and we proceed as in Proposition 3.7.4. For the new cases, we can check that $[\_,\_]$, $\hat{\_}$ and $\_ \circ H$ preserve n-totality.

- We can reify uniform total strategies as infinitary analytic proofs using the coalgebraic formulation as in Section 3.7: our definition reify is still of the required shape.
- We can show that $[\![\mathsf{reify}(\sigma)]\!] = \sigma$ by showing that, for each n and A, $\sigma_A =_n [\![\mathsf{reify}(\sigma_A)]\!]$. This is by lexicographic induction on

$$\langle n, \mathsf{tl}^+(\Gamma), \mathsf{hd}^+(\Gamma), \mathsf{tl}^-(\Gamma), \mathsf{hd}^-(\Gamma), \mathsf{L}(X,\Theta)\rangle$$

  and we can proceed as in Proposition 3.7.7. For the new cases, we can check that $[\_,\_]$, $\forall x.\_$ and $\_ \circ H$ respect $=_n$.
- We can show that $\mathsf{reify}([\![p]\!]) = p$ by showing that $\mathsf{reify} \circ [\![-]\!] = \mathbb{(}\, \alpha \,\mathbb{)} = \mathsf{id}$, as in Proposition 3.7.8. The remaining equations required to do so are verified in Proposition 4.3.8.

We can conclude that uniform total strategies on $[\![X;\Theta\vdash\Gamma]\!]$ are in bijective correspondence with infinitary analytic proofs of $X;\Theta\vdash\Gamma$, via the semantics. Thus given any proof p of $X;\Theta\vdash\Gamma$ in $\mathsf{WS1}$ we can compute $\mathsf{reify}([\![p]\!])$, which is the unique infinitary analytic proof whose semantics is $[\![p]\!]$. Thus, two proofs are semantically equivalent if and only if they have the same infinitary normal form.

Figure 4-5: Cut Elimination Procedure (wk_P for atoms and quantifiers)

$\mathsf{wk}_P(\mathsf{P}_{\neq})$	$=$	$\mathsf{P}_{\neq}$
$\mathsf{wk}_P(\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p,q))$	$=$	$\mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{wk}_P(p),\mathsf{wk}_P(q))$
$\mathsf{wk}_P(\mathsf{P}_{\mathsf{at}-}(p))$	$=$	$\mathsf{P}_{\mathsf{at}-}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}_{\mathsf{at}+}(p))$	$=$	$\mathsf{P}_{\mathsf{at}+}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}_{\forall}(p))$	$=$	$\mathsf{P}_{\forall}(\mathsf{wk}_P(p))$
$\mathsf{wk}_P(\mathsf{P}_{\exists}^{s}(p))$	$=$	$\mathsf{P}_{\exists}^{s}(\mathsf{wk}_P(p))$

4.5 Cut Elimination

In Sections 2.5.1 and 3.8.1 we defined a syntactic cut elimination procedure for analytic proofs. We now extend this to $\mathsf{WS1}$. Specifically, given an analytic proof $X;\Theta \vdash A,\Gamma,N^{\perp}$ and an analytic proof of $X;\Theta \vdash N,P$ we can construct an analytic proof of $X;\Theta \vdash A,\Gamma,P$.

4.5.1 Cut Elimination Procedure

As a starting point, we take the procedures defined in Figure 3-8 and note we can expand them to propagate $X;\Theta$ contexts additively. We hence only need to define cut and cut_2 on the new core proof rules.

To do this, we first define some (additional) auxiliary procedures.

- A procedure $\mathsf{wk}_{\overline{\psi}(\overrightarrow{s})}$, which takes a proof of $X;\Theta \vdash \Gamma$ and produces a proof of $X;\Theta,\overline{\psi}(\overrightarrow{s}) \vdash \Gamma$. This can be given using a trivial induction. Then $[\![\mathsf{wk}_{\overline{\psi}(\overrightarrow{s})}(p)]\!] = [\![p]\!]J_{\overline{\psi}(\overrightarrow{s})}$ where $J_{\overline{\psi}(\overrightarrow{s})} : \mathcal{M}_X^{\Theta,\overline{\psi}(\overrightarrow{s})} \to \mathcal{M}_X^{\Theta}$ is the inclusion functor.
- We can similarly give a procedure wk_x extracting a proof of $X \uplus \{x\};\Theta \vdash \Gamma$ from a proof of $X;\Theta \vdash \Gamma$. We have $[\![\mathsf{wk}_x(p)]\!] = [\![x]\!]U_x$ where $U_x : \mathcal{M}_{X\uplus\{x\}}^{\Theta} \to \mathcal{M}_X^{\Theta}$ forgets the valuation at x.
- We can define a substitution procedure. Suppose p is a proof of $X \uplus \{x\},\Theta \vdash \Gamma$ with free variables of s in X. Then we can give a proof p_x^s of $X,\Theta[s/x] \vdash \Gamma[s/x]$ by replacing all occurrences of x in p by s. We have $[\![p_x^s]\!] = [\![p]\!]\mathsf{set}_y^s$.

We then extend the procedures from Figure 3-8 to $\mathsf{WS1}$ in Figures 4-5, 4-6, 4-7. For termination, we need to use the fact that the weakening and substitution operations preserve the size of the proof, so that the termination argument given in Section 2.5.1 still applies.

4.5.2 Soundness

We next show that this procedure is sound with respect to the interpretation in $\mathcal{W}^{\mathcal{M}_X^{\Theta}}$.

Figure 4-6: Cut Elimination Procedure (rem_0 for atoms and quantifiers)

$$
\begin{array}{lcl}
\mathsf{rem}_0(\mathsf{P}_{\neq}) & = & \mathsf{P}_{\neq} \\
\mathsf{rem}_0(\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p,q)) & = & \mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{rem}_0(p),\mathsf{rem}_0(q)) \\
\mathsf{rem}_0(\mathsf{P}_{\mathsf{at}-}(p)) & = & \mathsf{P}_{\mathsf{at}-}(\mathsf{rem}_0(p)) \\
\mathsf{rem}_0(\mathsf{P}_{\mathsf{at}+}(p)) & = & \mathsf{P}_{\mathsf{at}+}(\mathsf{rem}_0(p)) \\
\mathsf{rem}_0(\mathsf{P}_{\forall}(p)) & = & \mathsf{P}_{\forall}(\mathsf{rem}_0(p)) \\
\mathsf{rem}_0(\mathsf{P}_{\exists}^{s}(p)) & = & \mathsf{P}_{\exists}^{s}(\mathsf{rem}_0(p))
\end{array}
$$

Figure 4-7: Cut Elimination Procedure (cut for atoms and quantifiers)

$$
\begin{array}{lcl}
\mathsf{cut} : X;\Theta \vdash A,\Gamma,N^{\perp} \times X;\Theta \vdash N,P & \to & X;\Theta \vdash A,\Gamma,P \\
\mathsf{cut}(\mathsf{P}_{\neq},\mathsf{P}_{\neq}) & = & \mathsf{P}_{\neq} \\
\mathsf{cut}(\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1,q_1),\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_2,q_2)) & = & \mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{cut}(p_1,p_2),\mathsf{cut}(q_1,q_2)) \\
\mathsf{cut}(\mathsf{P}_{\mathsf{at}-}(p),q) & = & \mathsf{P}_{\mathsf{at}-}(\mathsf{cut}(p,\mathsf{wk}_{\phi}(q))) \\
\mathsf{cut}(\mathsf{P}_{\mathsf{at}+}(p),q) & = & \mathsf{P}_{\mathsf{at}+}(\mathsf{cut}(p,q)) \\
\mathsf{cut}(\mathsf{P}_{\forall}(p),q) & = & \mathsf{P}_{\forall}(\mathsf{cut}(p,\mathsf{wk}_{x}(q))) \\
\mathsf{cut}(\mathsf{P}_{\exists}^{s}(p),q) & = & \mathsf{P}_{\exists}^{s}(\mathsf{cut}(p,q)) \\
 & & \\
\mathsf{cut}_2 : X;\Theta \vdash Q,\Gamma,N^{\perp} \times X;\Theta \vdash Q^{\perp},\Gamma^{\perp},P & \to & X;\Theta \vdash N^{\perp} \otimes P \\
\mathsf{cut}_2(\mathsf{P}_{\neq},\mathsf{P}_{\neq}) & = & \mathsf{P}_{\neq} \\
\mathsf{cut}_2(\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1,q_1),\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_2,q_2)) & = & \mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{cut}_2(p_1,p_2),\mathsf{cut}_2(q_1,q_2)) \\
\mathsf{cut}_2(\mathsf{P}_{\mathsf{at}+}(p),\mathsf{P}_{\mathsf{at}-}(q)) & = & \mathsf{cut}_2(p,q) \\
\mathsf{cut}_2(\mathsf{P}_{\exists}^{y}(p),\mathsf{P}_{\forall}(q)) & = & \mathsf{cut}_2(p,q_x^y)
\end{array}
$$

Figure 4-8: Diagram notation for elimination of cut in WS1

$$
\mathsf{cut}\ \dfrac{\dfrac{X \uplus \{x\};\Theta \vdash M,\Gamma,N^{\perp}}{X;\Theta \vdash \forall x.M,\Gamma,N^{\perp}} \qquad X;\Theta \vdash N,P}{X;\Theta \vdash \forall x.M,\Gamma,P}
$$

$$
\mapsto
$$

$$
\mathsf{cut}\ \dfrac{X \uplus \{x\};\Theta \vdash M,\Gamma,N^{\perp} \qquad \mathsf{wk}_x\ \dfrac{X;\Theta \vdash N,P}{X \uplus \{x\};\Theta \vdash N,P}}{\dfrac{X \uplus \{x\};\Theta \vdash M,\Gamma,P}{X;\Theta \vdash \forall x.M,\Gamma,P}}
$$

Note that $f \circ [\sigma, \tau] = [f \circ \sigma, f \circ \tau]$ and $[\sigma, \tau] \circ f = [\sigma \circ f, \tau \circ f]$. If f is strict, we have $f \circ [\sigma, \epsilon] = [f \circ \sigma, \epsilon]$. Finally, $\Lambda^{(-1)}([\sigma, \tau]) = [\Lambda^{(-1)}(\sigma), \Lambda^{(-1)}(\tau)]$.

Proposition 4.5.1 wk_P *is sound — that is, if* p *is a proof of* $\vdash A, \Gamma$ *then* $[\![\mathsf{wk}_P(p)]\!] = [\![\mathsf{P}^+_{\mathsf{wk}}(p)]\!]$. *(This is* $(\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![p]\!]$ *if* A *is negative and* $[\![p]\!] \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$ *if* A *is positive.).*

Proof We proceed by induction on p. For the cases where p is a rule of WS!, we can use Proposition 3.8.1 as $\mathcal{W}^{\mathcal{M}^{\ominus}_X}$ is a WS!-category. We only need to check the new cases, many of which are similar to those in Proposition 2.5.4.

- If $p = \mathsf{P}_{\neq}$ then the result holds trivially as the empty family is the only map of the required type.
- If $p = \mathsf{P}^{x,y,z}_{\mathsf{ma}}(p_1, p_2)$ then $[\![\mathsf{wk}(p)]\!] = [[\![\mathsf{wk}(p_1)]\!] H_{x,y,z}, [\![\mathsf{wk}(p_2)]\!]]$. If A is negative, then this is $[(\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![p_1]\!] H_{x,y,z}, (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![p_2]\!]] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [[\![p_1]\!] H_{x,y,z}, [\![p_2]\!]] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![p]\!]$. If A is positive we proceed similarly.
- If $p = \mathsf{P}_{\mathsf{at}-}(q)$ then $[\![\mathsf{wk}(p)]\!] = [\![\mathsf{P}_{\mathsf{at}-}(\mathsf{wk}(q))]\!] = [[\![\mathsf{wk}(q)]\!], \epsilon] = [(\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![q]\!], \epsilon] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [[\![q]\!], \epsilon] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![\mathsf{P}_{\mathsf{at}-(q)}]\!] = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![p]\!]$ as required.
- If $p = \mathsf{P}_{\mathsf{at}+}(q)$ then $[\![\mathsf{wk}(p)]\!] = [\![\mathsf{wk}(q)]\!] = [\![q]\!] \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af}) = [\![p]\!] \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$ as required.
- If $p = \mathsf{P}_{\forall}(q)$ then $[\![\mathsf{wk}(p)]\!] = \mathsf{dist}^{-1}_{\Gamma,P} \circ \widehat{[\![\mathsf{wk}(q)]\!]} = \mathsf{dist}^{-1}_{\Gamma,P} \circ (\widehat{(\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap}} \circ [\![q]\!])$. We must show that this is equal to $(\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ \mathsf{dist}^{-1}_{\Gamma} \circ \hat{[\![q]\!]}$ and so it is sufficient to show that $\mathsf{dist}_{\Gamma,P} \circ (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ \mathsf{dist}^{-1}_{\Gamma} \circ \hat{[\![q]\!]} = \widehat{((\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![q]\!])}$. By the universal property, it is sufficient to show that $\eta \circ U'_x(\mathsf{dist}_{\Gamma,P} \circ (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ \mathsf{dist}^{-1}_{\Gamma} \circ \hat{[\![q]\!]}) = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![q]\!]$. But the LHS is $\pi_x \circ \mathsf{dist}_{\Gamma,P} \circ (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ \mathsf{dist}^{-1}_{\Gamma} \circ \hat{[\![q]\!]} = (\mathsf{id} \multimap [\![\Gamma]\!]^-(\pi_x)) \circ (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ \mathsf{dist}^{-1}_{\Gamma} \circ \hat{[\![q]\!]} = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![\Gamma]\!]^-(\pi_x) \circ \mathsf{dist}^{-1}_{\Gamma} \circ \hat{[\![q]\!]} = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ \pi_x \circ \hat{[\![q]\!]} = (\mathsf{af} \multimap \mathsf{id}) \circ \mathsf{unit}^{-1}_{\multimap} \circ [\![q]\!]$ as required.
- If $p = \mathsf{P}^s_{\exists}(q)$ then $[\![\mathsf{wk}(p)]\!] = [\![\mathsf{P}^s_{\exists}(\mathsf{wk}(q))]\!] = [\![\mathsf{wk}(q)]\!] \circ \pi_s \circ \mathsf{dist}_{\Gamma,P} = [\![q]\!] \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af}) \circ \pi_s \circ \mathsf{dist}_{\Gamma,P} = [\![q]\!] \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af}) \circ [\![\Gamma, P]\!]^+(\pi_s) = [\![q]\!] \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af}) \circ ([\![\Gamma]\!]^+(\pi_s) \oslash \mathsf{id}) = [\![q]\!] \circ [\![\Gamma]\!]^+(\pi_s) \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af}) = [\![p]\!] \circ \mathsf{unit}_{\oslash} \circ (\mathsf{id} \oslash \mathsf{af})$ as required. ∎

Proposition 4.5.2 rem_0 *is sound — that is, if* p *is a proof of* $\vdash A, \Gamma$ *then* $[\![\mathsf{rem}_0(p)]\!] = \mathsf{unit}_{\multimap} \circ [\![p]\!]$ *if* A *is negative and* $[\![p]\!] \circ \mathsf{unit}^{-1}_{\oslash}$ *if* A *is positive.*

Proof We proceed by induction on p. For the cases where p is a rule of $\mathsf{WS!}$, we can use Proposition 3.8.1 as $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a WS!-category. We only need to check the new cases, many of which are similar to those in Proposition 2.5.2.

- If $p = \mathsf{P}_{\neq}$ then the result holds trivially as the empty family is the only map of the required type.
- If $p = \mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1, p_2)$ then $[\![\mathsf{rem}_0(p)]\!] = [[\![\mathsf{rem}_0(p_1)]\!]H_{x,y,z}, [\![\mathsf{rem}_0(p_2)]\!]]$. If A is negative, then this is $[\mathsf{unit}_{\multimap} \circ [\![p_1]\!]H_{x,y,z}, \mathsf{unit}_{\multimap} \circ [\![p_2]\!]] = \mathsf{unit}_{\multimap} \circ [[\![p_1]\!]H_{x,y,z}, [\![p_2]\!]] = \mathsf{unit}_{\multimap} \circ [\![p]\!]$. If A is positive we proceed similarly.
- If $p = \mathsf{P}_{\mathsf{at}-}(q)$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}_{\mathsf{at}-}(\mathsf{rem}_0(q))]\!] = [[\![\mathsf{rem}_0(q)]\!], \epsilon] = [\mathsf{unit}_{\multimap} \circ [\![q]\!], \epsilon] = \mathsf{unit}_{\multimap} \circ [[\![q]\!], \epsilon] = \mathsf{unit}_{\multimap} \circ [\![\mathsf{P}_{\mathsf{at}-(q)}]\!] = \mathsf{unit}_{\multimap} \circ [\![p]\!]$ as required.
- If $p = \mathsf{P}_{\mathsf{at}+}(q)$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{rem}_0(q)]\!] = [\![q]\!] \circ \mathsf{unit}_{\oslash}^{-1} = [\![p]\!] \circ \mathsf{unit}_{\oslash}^{-1}$ as required.
- If $p = \mathsf{P}_{\forall}(q)$ then $[\![\mathsf{rem}_0(p)]\!] = \mathsf{dist}_\Gamma^{-1} \circ \widehat{[\![\mathsf{rem}_0(q)]\!]} = \mathsf{dist}_\Gamma^{-1} \circ (\widehat{\mathsf{unit}_{\multimap} \circ [\![q]\!]})$. We must show that this is equal to $\mathsf{unit}_{\multimap} \circ \mathsf{dist}_{\Gamma,\mathbf{0}}^{-1} \circ \hat{[\![q]\!]}$ and so it is sufficient to show that $\mathsf{dist}_\Gamma \circ \mathsf{unit}_{\multimap} \circ \mathsf{dist}_{\Gamma,\mathbf{0}}^{-1} \circ \hat{[\![q]\!]} = (\widehat{\mathsf{unit}_{\multimap} \circ [\![q]\!]})$. By the universal property, it is sufficient to show that $\eta \circ U'_x(\mathsf{dist}_\Gamma \circ \mathsf{unit}_{\multimap} \circ \mathsf{dist}_{\Gamma,\mathbf{0}}^{-1} \circ \hat{[\![q]\!]}) = \mathsf{unit}_{\multimap} \circ [\![q]\!]$. But the LHS is $\pi_x \circ \mathsf{dist}_\Gamma \circ \mathsf{unit}_{\multimap} \circ \mathsf{dist}_{\Gamma,\mathbf{0}}^{-1} \circ \hat{[\![q]\!]} = [\![\Gamma]\!]^-(\pi_x) \circ \mathsf{unit}_{\multimap} \circ \mathsf{dist}_{\Gamma,\mathbf{0}}^{-1} \circ \hat{[\![q]\!]} = \mathsf{unit}_{\multimap} \circ (\mathsf{id} \multimap [\![\Gamma]\!]^-(\pi_x)) \circ \mathsf{dist}_{\Gamma,\mathbf{0}}^{-1} \circ \hat{[\![q]\!]} = \mathsf{unit}_{\multimap} \circ ([\![\Gamma, \mathbf{0}]\!]^-(\pi_x)) \circ \mathsf{dist}_{\Gamma,\mathbf{0}}^{-1} \circ \hat{[\![q]\!]} = \mathsf{unit}_{\multimap} \circ \pi_x \circ \hat{[\![q]\!]} = \mathsf{unit}_{\multimap} \circ [\![q]\!]$ as required.
- If $p = \mathsf{P}_{\exists}^s(q)$ then $[\![\mathsf{rem}_0(p)]\!] = [\![\mathsf{P}_{\exists}^s(\mathsf{rem}_0(q))]\!] = [\![\mathsf{rem}_0(q)]\!] \circ \pi_s \circ \mathsf{dist}_\Gamma = [\![q]\!] \circ \mathsf{unit}_{\oslash}^{-1} \circ \pi_s \circ \mathsf{dist}_\Gamma = [\![q]\!] \circ \mathsf{unit}_{\oslash}^{-1} \circ [\![\Gamma]\!]^+(\pi_s) = [\![q]\!] \circ ([\![\Gamma]\!]^+(\pi_s) \oslash \mathsf{id}) \circ \mathsf{unit}_{\oslash}^{-1} = [\![q]\!] \circ [\![\Gamma, \mathbf{0}]\!]^+(\pi_s) \circ \mathsf{unit}_{\oslash}^{-1} = [\![p]\!] \circ \mathsf{unit}_{\oslash}^{-1}$ as required. ∎

Proposition 4.5.3 *If p_1 is a proof of $\vdash A, \Gamma, N^\perp$ and p_2 is a proof of $\vdash N, R$ then $[\![\mathsf{cut}(p_1, p_2)]\!] = [\![\mathsf{P}_{\mathsf{cut}}(p_1, p_2)]\!]$. That is,*

- $[\![\mathsf{cut}(p_1, p_2)]\!] = \Lambda_I(\Lambda_I^{-1}([\![p_1]\!]) \circ \Lambda_I^{-1}([\![p_2]\!]))$ *if A is negative.*
- $[\![\mathsf{cut}(p_1, p_2)]\!] = [\![p_1]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}[\![p_2]\!])$ *if A is positive.*
- $[\![\mathsf{cut}_2(p_1, p_2)]\!] = [\![p_1]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![p_2]\!])$.

Proof Since $\mathcal{W}^{\mathcal{M}_X^\Theta}$ is a WS!-category, we can use Proposition 3.8.1 for all of the cases from $\mathsf{WS!}$. We thus only need to check the new cases. For cut:

- For sequents with a negative head, $[\![\mathsf{cut}(\mathsf{P}_{\neq}, \mathsf{P}_{\neq})]\!] = \Lambda_I(\Lambda_I^{-1}([\![\mathsf{P}_{\neq}]\!]) \circ \Lambda_I^{-1}([\![\mathsf{P}_{\neq}]\!]))$ as there is only one map of the required type as the family is empty. Similar for sequents with a positive head.
- For sequents with a negative head, $[\![\mathsf{cut}(\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1, q_1), \mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_2, q_2))]\!] = [\![\mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{cut}(p_1, p_2), \mathsf{cut}(q_1, q_2))]\!] = [[\![\mathsf{cut}(p_1, p_2)]\!]H, [\![\mathsf{cut}(q_1, q_2)]\!]] = [\Lambda_I(\Lambda_I^{-1}([\![p_1]\!]) \circ$

$\Lambda_I^{-1}([\![p_2]\!]))H, \Lambda_I(\Lambda_I^{-1}([\![q_1]\!]) \circ \Lambda_I^{-1}([\![q_2]\!]))] = \Lambda_I[\Lambda_I^{-1}([\![p_1]\!]) \circ \Lambda_I^{-1}([\![p_2]\!])H, \Lambda_I^{-1}([\![q_1]\!]) \circ \Lambda_I^{-1}([\![q_2]\!])] = \Lambda_I[\Lambda_I^{-1}([\![p_1]\!])H, \Lambda_I^{-1}([\![q_1]\!])] \circ [\Lambda_I^{-1}([\![p_2]\!])H, \Lambda_I^{-1}([\![q_2]\!])] = \Lambda_I(\Lambda_I^{-1}[[\![p_1]\!]H, [\![q_1]\!]] \circ \Lambda_I^{-1}[[\![p_2]\!]H, [\![q_2]\!]]) = \Lambda_I(\Lambda_I^{-1}([\![\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1, q_1)]\!]) \circ \Lambda_I^{-1}([\![\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_2, q_2)]\!]))$ as required. Similar for sequents with a positive head.

- $[\![\mathsf{cut}(\mathsf{P}_{\mathsf{at}-}(p), q)]\!] = [\![\mathsf{P}_{\mathsf{at}-}(\mathsf{cut}(p, \mathsf{wk}_\phi(q)))]\!] = [[\![\mathsf{cut}(p, \mathsf{wk}_\phi(q))]\!], \epsilon]$
$= [\Lambda_I(\Lambda_I^{-1}([\![p]\!]) \circ \Lambda_I^{-1}([\![\mathsf{wk}_\phi(q)]\!])), \epsilon]$
$= [\Lambda_I(\Lambda_I^{-1}([\![p]\!]) \circ \Lambda_I^{-1}([\![\mathsf{wk}_\phi(q)]\!])), \Lambda_I(\Lambda_I^{-1}(\epsilon) \circ \Lambda_I^{-1}([\![\mathsf{wk}_\phi(q)]\!]))]$
$= \Lambda_I[\Lambda_I^{-1}([\![p]\!]) \circ \Lambda_I^{-1}([\![\mathsf{wk}_\phi(q)]\!]), \Lambda_I^{-1}(\epsilon) \circ \Lambda_I^{-1}([\![\mathsf{wk}_\phi(q)]\!])]$
$= \Lambda_I[\Lambda_I^{-1}([\![p]\!]), \Lambda_I^{-1}(\epsilon)] \circ \Lambda_I^{-1}([\![\mathsf{wk}_\phi(q)]\!])$
$= \Lambda_I(\Lambda_I^{-1}([[\![p]\!], \epsilon] \circ \Lambda_I^{-1}([\![\mathsf{wk}_\phi(q)]\!])$
$= \Lambda_I(\Lambda_I^{-1}([\![\mathsf{P}_{\mathsf{at}-(p)}]\!]) \circ \Lambda_I^{-1}([\![q]\!]))$ as required.
- $[\![\mathsf{cut}(\mathsf{P}_{\mathsf{at}+}(p), q)]\!] = [\![\mathsf{P}_{\mathsf{at}+}(\mathsf{cut}(p, q))]\!] = [\![\mathsf{cut}(p, q)]\!] = [\![p]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!]) = [\![\mathsf{P}_{\mathsf{at}+}(p)]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!])$ as required.
- $[\![\mathsf{cut}(\mathsf{P}_\forall(p), q)]\!] = [\![\mathsf{P}_\forall(\mathsf{cut}(p, \mathsf{wk}_x(q)))]\!] = \mathsf{dist}_{\Gamma, N^\perp}^{-1} \circ \widehat{[\![\mathsf{cut}(p, \mathsf{wk}_x(q))]\!]} = \mathsf{dist}_{\Gamma, N^\perp}^{-1} \circ \Lambda_I(\Lambda_I^{-1}(\widehat{[\![p]\!]) \circ \Lambda_I^{-1}}(U'x([\![q]\!]))$. We need to show that this is equal to $\Lambda_I(\Lambda_I^{-1}(\mathsf{dist}_{\Gamma,P}^{-1} \circ \hat{[\![p]\!]}) \circ \Lambda_I^{-1}([\![q]\!]))$ and so it is sufficient to show that $\mathsf{dist}_{\Gamma,N^\perp} \circ \Lambda_I(\Lambda_I^{-1}(\mathsf{dist}_{\Gamma,P}^{-1} \circ \hat{[\![p]\!]}) \circ \Lambda_I^{-1}([\![q]\!])) = \Lambda_I(\Lambda_I^{-1}([\![p]\!]) \widehat{\circ \Lambda_I^{-1}}(U'x([\![q]\!])))$ and by the universal property it is sufficient to show that $\eta \circ U'_x(\mathsf{dist}_{\Gamma,N^\perp} \circ \Lambda_I(\Lambda_I^{-1}(\mathsf{dist}_{\Gamma,P}^{-1} \circ \hat{[\![p]\!]}) \circ \Lambda_I^{-1}([\![q]\!]))) = \Lambda_I(\Lambda_I^{-1}([\![p]\!]) \circ \Lambda_I^{-1}(U'x([\![q]\!])))$. The LHS is $\pi_x \circ \mathsf{dist}_{\Gamma,N^\perp} \circ \Lambda_I(\Lambda_I^{-1}(\mathsf{dist}_{\Gamma,P}^{-1} \circ \hat{[\![p]\!]}) \circ \Lambda_I^{-1}([\![q]\!])) = [\![\Gamma, N^\perp]\!]^-(\pi_x) \circ \Lambda_I(\Lambda_I^{-1}(\mathsf{dist}_{\Gamma,P}^{-1} \circ \hat{[\![p]\!]}) \circ \Lambda_I^{-1}([\![q]\!])) = \Lambda_I([\![\Gamma]\!]^-(\pi_x) \circ \Lambda_I^{-1}(\mathsf{dist}_{\Gamma,P}^{-1} \circ \hat{[\![p]\!]}) \circ \Lambda_I^{-1}([\![q]\!])) = \Lambda_I(\Lambda_I^{-1}(\pi_x \circ \hat{[\![p]\!]}) \circ \Lambda_I^{-1}([\![q]\!])) = \Lambda_I(\Lambda_I^{-1}(\pi_x \circ \hat{[\![p]\!]}) \circ \Lambda_I^{-1}(U'_x([\![q]\!])))$ as required.
- $[\![\mathsf{cut}(\mathsf{P}_\exists^s(p), q)]\!] = [\![\mathsf{P}_\exists^s(\mathsf{cut}(p, q))]\!] = [\![\mathsf{cut}(p, q)]\!] \circ \pi_s \circ \mathsf{dist}_{\Gamma,P} = [\![p]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!])) \circ \pi_s \circ \mathsf{dist}_{\Gamma,P} = [\![p]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!])) \circ [\![\Gamma, P]\!]^+(\pi_s) = [\![p]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!])) \circ ([\![\Gamma]\!]^+(\pi_s) \oslash \mathsf{id}) = [\![p]\!] \circ ([\![\Gamma]\!]^+(\pi_s) \oslash \mathsf{id}) \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!])) = [\![p]\!] \circ [\![\Gamma, N^\perp]\!]^+(\pi_s) \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!])) = [\![p]\!] \circ \pi_s \circ \mathsf{dist}_{\Gamma,N^\perp} \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!])) = [\![\mathsf{P}_\exists^s(p)]\!] \circ (\mathsf{id} \oslash \Lambda_I^{-1}([\![q]\!]))$ as required.

For cut_2:

- $[\![\mathsf{cut}(\mathsf{P}_\neq, \mathsf{P}_\neq)]\!] = \Lambda_I(\Lambda_I^{-1}([\![\mathsf{P}_\neq]\!]) \circ \Lambda_I^{-1}([\![\mathsf{P}_\neq]\!]))$ as there is only one map of the required type as the family is empty.
- $[\![\mathsf{cut}_2(\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1, q_1), \mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_2, q_2))]\!] = [\![\mathsf{P}_{\mathsf{ma}}^{x,y,z}(\mathsf{cut}_2(p_1, p_2), \mathsf{cut}_2(q_1, q_2))]\!] = [[\![\mathsf{cut}_2(p_1, p_2)]\!]H, [\![\mathsf{cut}_2(q_1, q_2)]\!]] = [[\![p_1]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![p_2]\!])H, [\![q_1]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![q_2]\!])] = [[\![p_1]\!]H, [\![q_1]\!]] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[[\![p_2]\!]H, [\![q_2]\!]]) = [\![\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_1, q_1)]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ \Lambda_I^{-1}([\![\mathsf{P}_{\mathsf{ma}}^{x,y,z}(p_2, q_2)]\!])$ as required.
- $[\![\mathsf{cut}_2(\mathsf{P}_{\mathsf{at}+}(p), \mathsf{P}_{\mathsf{at}-}(q))]\!] = [\![\mathsf{cut}_2(p, q)]\!] = [\![p]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![q]\!]) = [\![\mathsf{P}_{\mathsf{at}+(p)}]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}[\![\mathsf{P}_{\mathsf{at}-(q)}]\!])$ as required. We know that $[\![q]\!] = [\![\mathsf{Pat}-(q)]\!]$ as we

must have $\overline{\phi} \in \Theta$ for the appropriate atom and so $[[\![q]\!], \epsilon]_{\overline{\phi},\phi} = [\![q]\!]$.

- $[\![\mathsf{cut}_2(\mathsf{P}^s_\exists(p), \mathsf{P}_\forall(q))]\!] = [\![\mathsf{cut}_2(p, q^s_x)]\!] = [\![p]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}([\![q^s_x]\!])) = [\![p]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{set}'^x_s([\![q]\!])))$. We must show that this is equal to $[\![p]\!] \circ \pi_s \circ \mathsf{dist}_{\Gamma,N^\perp} \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{dist}^{-1}_{\Gamma,P} \circ \hat{[\![q]\!]})$. The latter is $[\![p]\!] \circ ([\![\Gamma]\!]^+(\pi_s) \oslash \mathsf{id}) \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{dist}^{-1}_{\Gamma,P} \circ \hat{[\![q]\!]}) = [\![p]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes [\![\Gamma]\!]^+(\pi_s) \circ \Lambda_I^{-1}(\mathsf{dist}^{-1}_{\Gamma,P} \circ \hat{[\![q]\!]})) = [\![p]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\pi_s \circ \hat{[\![q]\!]})) = [\![p]\!] \circ \mathsf{wk} \circ \mathsf{sym} \circ (\mathsf{id} \otimes \Lambda_I^{-1}(\mathsf{set}'^x_s([\![q]\!])))$ as required. ∎

4.6 Axioms

As well as using WS1 as a general first-order logic, we can introduce rules and axioms based on a specific intended model. For example, if we fix $\mathcal{L}$ to be a language about natural numbers (with constants including zero 0, and successor s) we can introduce an induction rule:

$$\frac{X; \Theta \vdash N[0/x] \qquad X \uplus \{x\}; \Theta \vdash N[s(x)/x], N^\perp}{X; \Theta \vdash \forall x.N}$$

We can interpret formulas as families of games varying over the Θ-satisfying valuation only, and proofs as families of winning strategies — there is an evident interpretation of the above rule. We will next use these ideas as a tool for giving specifications on programs in WS1.

Chapter 5

Programs and their Properties

In this chapter we embed imperative total programming languages into an extension of WS1 with an infinite choice operator. Example programs include objects representing a postfix calculator and a data-independent set. We show how we can use our logic to represent properties of these programs.

In this chapter we shall demonstrate the expressivity of WS1 with respect to programs and properties upon them. As described in Section 3.4.3, we can interpret recursion-free Idealized Algol over finitary data types in WS1, using the embedding of Intuitionistic Linear Logic and the Boolean cell described in Figure 3-3. In this chapter we will make this embedding explicit, and show how further features can be embedded, such as:

- Call-by-value
- Coroutines
- Infinitely deep state, e.g. stacks
- Natural numbers
- For loops
- Data-independent ground types.

We will also use the first-order structure in WS1 to represent a large collection of program properties as formulas of WS1.

First, we will recall the standard embeddings of CBN and CBV lambda calculi into LLP. By composing these embeddings with the embedding of LLP inside WS1 given in Section 3.4.4, we obtain an embedding of these calculi into WS1.

We can extend the call-by-value calculus with imperative features: ground store, coroutines, and an encaps operator which reflects the anamorphism rule of WS1. The

latter can be used to represent a limited form of higher-order store and some infinite data structures (e.g. a stack of Booleans, following Section 3.4.3). We extend the CBV embedding into WS1 with these features.

We next extend WS1 with a game representing natural numbers: this is straightforward, and the results from previous chapters can also be extended. Using this, we give an embedding of a call-by-name imperative total programming language with a natural number ground type in our logic. This language contains the imperative features above, and all primitive recursive functions can be defined. The language is expressive: for example, one can define a stateful object representing an (arbitrarily large) set of natural numbers. Thus we see that the programming expressivity of WS1 is high.

We then show how the first-order structure can be used to formulate a large range of behavioural properties on these programs. Finally, we show how the first-order structure can be used in a different way: by relating each unary predicate to an atomic type, we can write programs that are independent of the underlying ground types (e.g. a data-independent set).

5.1 Finitary Lambda Calculi

In this section, we will show how finitary call-by-name and call-by-value lambda calculi can be embedded inside WS1. We achieve this by composing their standard embeddings into LLP with the embedding of LLP inside WS! given in in Section 3.4.4.

5.1.1 A Calculus of Finite Types

We first describe a calculus of finite types. We can equip this calculus with call-by-name or call-by-value operational semantics at will.

The types of λ_{fin} are as follows:

$$T := \quad 1 \quad | \quad 2 \quad | \quad T \times T \quad | \quad T \to T$$

A typing judgement of λ_{fin} is of the form $x_1 : T_1, \ldots, x_n : T_n \vdash s : T$ where T_i,T range over terms, x_i over variables and s over terms. The well-typed terms of λ_{fin} are defined in Figure 5-1.

We can give call-by-name and call-by-value operational semantics to λ_{fin} in a standard manner.

Figure 5-1: Terms of λ_{fin}

$$\frac{}{x_1 : T_1, \ldots, x_n : T_n \vdash x_i : T_i} \qquad \frac{\Gamma, x : S \vdash t : T}{\Gamma \vdash \lambda x.t : S \to T} \qquad \frac{\Gamma \vdash f : S \to T \qquad \Gamma \vdash s : S}{\Gamma \vdash fs : T}$$

$$\frac{\Gamma \vdash s : S \times T}{\Gamma \vdash \pi_1(s) : S} \qquad \frac{\Gamma \vdash s : S \times T}{\Gamma \vdash \pi_2(s) : T} \qquad \frac{\Gamma \vdash s : S \qquad \Gamma \vdash t : T}{\Gamma \vdash \langle s, t \rangle : S \times T}$$

$$\frac{}{\Gamma \vdash \mathtt{tt} : 2} \qquad \frac{}{\Gamma \vdash \mathtt{ff} : 2} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash f : A}{\Gamma, b : 2 \vdash \mathsf{case}(b, t, f) : A}$$

$$\frac{}{\Gamma \vdash u : 1}$$

5.1.2 Call-by-name Lambda Calculus

From CBN to LLP: Types

A map ϕ^- embedding call-by-name λ_{fin} into LLP was given in [53]. We recall this embedding here. Types are mapped to negative formulas of LLP, with:

- $\phi^-(1) = ?\mathbf{1}$
- $\phi^-(2) = ?\mathbf{1} \parr ?\mathbf{1}$
- $\phi^-(A \times B) = \phi^-(A) \& \phi^-(B)$
- $\phi^-(A \to B) = ?\phi^-(A)^\perp \parr \phi^-(B)$.

Note that no linear lifts are used here, so the resulting LLP formulas are reusable.

From CBN to LLP: Terms

A term $x_1 : T_1, \ldots, x_n : T_n \vdash s : T$ is translated to an LLP proof of

$$\vdash ?\phi^-(T_1)^\perp, \ldots, ?\phi^-(T_n)^\perp, \phi^-(T).$$

- Variables and abstraction:

$$\frac{\dfrac{\dfrac{}{\vdash \phi^-(T_i), \phi^-(T_i)^\perp}}{\vdash \phi^-(T_i), ?\phi^-(T_i)^\perp}}{\vdash \phi^-(T_i), ?\phi^-(T_1)^\perp, \ldots, ?\phi^-(T_n)^\perp} \qquad \frac{\vdash ?\phi^-(\Gamma)^\perp, ?\phi^-(S)^\perp, \phi^-(T)}{\vdash ?\phi^-(\Gamma)^\perp, ?\phi^-(S)^\perp \parr \phi^-(T)}$$

- Application:

$$\dfrac{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(S) \otimes \phi^-(T) \qquad \dfrac{\dfrac{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(S)}{\vdash ?\phi^-(\Gamma)^\perp, !\phi^-(S)} \qquad \dfrac{}{\vdash \phi^-(T)^\perp, \phi^-(T)}}{\vdash !\phi^-(S) \otimes \phi^-(T)^\perp, ?\phi^-(\Gamma)^\perp, \phi^-(T)}}{\dfrac{\vdash ?\phi^-(\Gamma)^\perp, ?\phi^-(\Gamma)^\perp, \phi^-(T)}{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(T)}}$$

- Pairing and projection:

$$\dfrac{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(S) \qquad \vdash ?\phi^-(\Gamma)^\perp, \phi^-(T)}{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(S) \& \phi^-(T)}$$

$$\dfrac{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(S) \& \phi^-(T) \qquad \dfrac{\dfrac{}{\vdash \phi^-(S)^\perp, \phi^-(S)}}{\vdash \phi^-(S)^\perp \oplus \phi^-(T)^\perp, \phi^-(S)}}{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(S)}$$

- Boolean true, false; unit:

$$\dfrac{\dfrac{\dfrac{}{\vdash ?\phi^-(\Gamma)^\perp, \mathbf{1}, ?\mathbf{1}}}{\vdash ?\phi^-(\Gamma)^\perp, ?\mathbf{1}, ?\mathbf{1}}}{\vdash ?\phi^-(\Gamma)^\perp, ?\mathbf{1} \parr ?\mathbf{1}} \qquad \dfrac{\dfrac{\dfrac{}{\vdash ?\phi^-(\Gamma)^\perp, ?\mathbf{1}, \mathbf{1}}}{\vdash ?\phi^-(\Gamma)^\perp, ?\mathbf{1}, ?\mathbf{1}}}{\vdash ?\phi^-(\Gamma)^\perp, ?\mathbf{1} \parr ?\mathbf{1}} \qquad \dfrac{\dfrac{}{\vdash ?\phi^-(\Gamma)^\perp, \mathbf{1}}}{\vdash ?\phi^-(\Gamma)^\perp, ?\mathbf{1}}$$

- Case:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(A)}{\vdash ?\phi^-(\Gamma)^\perp, \perp, \phi^-(A)}}{\vdash ?\phi^-(\Gamma)^\perp, !\perp, \phi^-(A)} \qquad \dfrac{\dfrac{\vdash ?\phi^-(\Gamma)^\perp, \phi^-(A)}{\vdash ?\phi^-(\Gamma)^\perp, \perp, \phi^-(A)}}{\vdash ?\phi^-(\Gamma)^\perp, !\perp, \phi^-(A)}}{\vdash ?\phi^-(\Gamma)^\perp, ?\phi^-(\Gamma)^\perp, !\perp \otimes !\perp, \phi^-(A), \phi^-(A)}}{\dfrac{\vdash ?\phi^-(\Gamma)^\perp, ?\phi^-(\Gamma)^\perp, ?(!\perp \otimes !\perp), \phi^-(A), \phi^-(A)}{\vdash ?\phi^-(\Gamma)^\perp, ?(!\perp \otimes !\perp), \phi^-(A)}}$$

Using this translation, call-by-name reduction of lambda terms can be simulated using cut elimination of LLP [53].

From CBN to WS

The composition of the above to translations give an interpretation $i \circ \phi^-$ of CBN inside WS1. Here we discuss the result of this translation.

First, we note that if a type is constructed without using $\times$, the resulting family in WS1 is singleton (this can be shown by an easy induction on types).

The type of Booleans 2 is translated to the LLP formula $?\mathbf{1} \parr ?\mathbf{1}$. This is mapped to the WS singleton family $?(\top \oslash \mathbf{1}) \parr ?(\top \oslash \mathbf{1})$, which is isomorphic to $?\top \parr ?\top \cong \top \oplus \top$.

A term of type 2 is translated to an LLP proof of $\vdash ?\mathbf{1} \parr ?\mathbf{1}$. This is translated to a WS proof of $\vdash \bot, \top \oplus \top$, modulo the simplification above.

The type $2 \times 2 \to 2$ is translated to LLP as the formula $?(!\bot \otimes !\bot) \parr ?\mathbf{1} \parr ?\mathbf{1}$. This is mapped to the WS singleton $?(\top \oslash (!\bot \otimes !\bot)) \parr ?\top \parr ?\top$. A term of type $2 \times 2 \to 2$ is translated to an LLP proof of $\vdash ?(!\bot \otimes !\bot) \parr ?\mathbf{1} \parr ?\mathbf{1}$. This is translated to a WS proof of $\vdash \bot, ?(\top \oslash (!\bot \otimes !\bot)) \parr ?\top \parr ?\top$. By applying isomorphisms, this is equivalent to a proof of $\vdash \bot, ?(\top \oslash (\bot \& \bot)), \top \oplus \top$ or $\vdash \bot \lhd (\top \oplus \top), ?(\top \oslash (\bot \& \bot))$, which is $!\mathbf{B} \multimap \mathbf{B}$ and agrees with the Intuitionistic Linear Logic transformation.

5.1.3 Call-by-value Lambda Calculus

From CBV to LLP: Types

We describe a map ϕ^+ embedding call-by-value λ_{fin} into LLP, following [53]. Types are mapped to positive formulas of LLP, with

- $\phi^+(1) = \mathbf{1}$
- $\phi^+(2) = \mathbf{1} \oplus \mathbf{1}$
- $\phi^+(A \times B) = \phi^+(A) \otimes \phi^+(B)$
- $\phi^+(A \to B) = !(\phi^+(A)^\perp \parr \uparrow \phi^+(B))$.

In [53], the embedding is slightly different: ? is used rather than $\uparrow$ in the translation of $\to$. This allows continuation control operators to be modelled (the $\lambda\mu$-calculus). We chose the simpler setting as it will allow us to express a different operator (encaps) which is closely related to the anamorphism rule of our logic. To obtain the original presentation, all uses of $\uparrow$ (resp. $\downarrow$) can be typographically replaced for ? (resp. !) in this embedding.

Note that for all T, $\phi^+(T)^\perp$ is reusable.

From CBV to LLP: Terms

A term $x_1 : T_1, \ldots, x_n : T_n \vdash s : T$ is translated to an LLP proof of

$$\vdash \uparrow \phi^+(T), \phi^+(T_1)^\perp, \ldots, \phi^+(T_n)^\perp.$$

- Variables and abstraction:

$$\dfrac{\dfrac{}{\vdash \phi^+(T_i), \phi^+(T_i)^\perp}}{\dfrac{\vdash \uparrow \phi^+(T_i), \phi^+(T_i)^\perp}{\vdash \uparrow \phi^+(T_i), \phi^+(T_1)^\perp, \ldots, \phi^+(T_n)^\perp}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \uparrow \phi^+(T), \phi^+(S)^\perp, \phi^+(\Gamma)^\perp}{\vdash \phi^+(S)^\perp, \uparrow \phi^+(T), \phi^+(\Gamma)^\perp}}{\vdash \phi^+(S)^\perp \parr \uparrow \phi^+(T), \phi^+(\Gamma)^\perp}}{\vdash !(\phi^+(S)^\perp \parr \uparrow \phi^+(T)), \phi^+(\Gamma)^\perp}}{\vdash \uparrow !(\phi^+(S)^\perp \parr \uparrow \phi^+(T)), \phi^+(\Gamma)^\perp}$$

- Application:

$$\dfrac{\vdash \uparrow !(\phi^+(S)^\perp \parr \uparrow \phi^+(T)), \phi^+(\Gamma)^\perp \qquad \dfrac{\dfrac{\vdash \uparrow \phi^+(S), \phi^+(\Gamma)^\perp \qquad q}{\vdash ?(\phi^+(S) \otimes \downarrow \phi^+(T)^\perp), \uparrow \phi^+(T), \phi^+(\Gamma)^\perp}}{\vdash \downarrow ?(\phi^+(S) \otimes \downarrow \phi^+(T)), \uparrow \phi^+(T), \phi^+(\Gamma)^\perp}}{\dfrac{\vdash \uparrow \phi^+(T), \phi^+(\Gamma)^\perp, \phi^+(\Gamma)^\perp}{\vdash \uparrow \phi^+(T), \phi^+(\Gamma)^\perp}}$$

  where q is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \phi^+(S), \phi^+(S)^\perp} \qquad \dfrac{}{\vdash \uparrow \phi^+(T), \downarrow \phi^+(T)^\perp}}{\vdash \phi^+(S) \otimes \downarrow \phi^+(T)^\perp, \uparrow \phi^+(T), \phi^+(S)^\perp}}{\vdash ?(\phi^+(S) \otimes \downarrow \phi^+(T)^\perp), \uparrow \phi^+(T), \phi^+(S)^\perp}}{\vdash ?(\phi^+(S) \otimes \downarrow \phi^+(T)^\perp), \uparrow \phi^+(T), \downarrow \phi^+(S)^\perp}$$

- Pairing and projection:

$$\dfrac{\dfrac{\vdash \uparrow \phi^+(S), \phi^+(\Gamma)^\perp \qquad \dfrac{\dfrac{\vdash \uparrow \phi^+(T), \phi^+(\Gamma)^\perp \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \phi^+(S), \phi^+(S)^\perp} \qquad \dfrac{}{\vdash \phi^+(T), \phi^+(T)^\perp}}{\vdash \phi^+(S) \otimes \phi^+(T), \phi^+(S)^\perp, \phi^+(T)^\perp}}{\vdash \uparrow (\phi^+(S) \otimes \phi^+(T)), \phi^+(S)^\perp, \phi^+(T)^\perp}}{\vdash \uparrow (\phi^+(S) \otimes \phi^+(T)), \phi^+(S)^\perp, \downarrow \phi^+(T)^\perp}}{\vdash \uparrow (\phi^+(S) \otimes \phi^+(T)), \phi^+(S)^\perp, \phi^+(\Gamma)^\perp}}{\vdash \uparrow (\phi^+(S) \otimes \phi^+(T)), \phi^+(S)^\perp, \downarrow \phi^+(S)^\perp, \phi^+(\Gamma)^\perp}}{\vdash \uparrow (\phi^+(S) \otimes \phi^+(T)), \phi^+(\Gamma)^\perp, \phi^+(\Gamma)^\perp}}{\vdash \uparrow (\phi^+(S) \otimes \phi^+(T)), \phi^+(\Gamma)^\perp}$$

$$\dfrac{\vdash \uparrow (\phi^-(S) \otimes \phi^-(T)), \phi^-(\Gamma)^\perp \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \phi^-(S), \phi^-(S)}}{\vdash \uparrow \phi^-(S), \phi^-(S)}}{\vdash \uparrow \phi^-(S), \phi^-(S), \phi^-(T)}}{\vdash \uparrow \phi^-(S), \phi^-(S) \parr \phi^-(T)}}{\vdash \uparrow \phi^-(S), \downarrow (\phi^-(S) \parr \phi^-(T))}}{\vdash \uparrow \phi^-(S), \phi^-(\Gamma)^\perp}$$

- Boolean true, false; unit:

$$\oplus_1 \frac{\dfrac{}{\vdash \mathbf{1}, \phi^-(\Gamma)^\perp}}{\dfrac{\vdash \mathbf{1} \oplus \mathbf{1}, \phi^-(\Gamma)^\perp}{\vdash \uparrow (\mathbf{1} \oplus \mathbf{1}), \phi^-(\Gamma)^\perp}} \qquad \oplus_2 \frac{\dfrac{}{\vdash \mathbf{1}, \phi^-(\Gamma)^\perp}}{\dfrac{\vdash \mathbf{1} \oplus \mathbf{1}, \phi^-(\Gamma)^\perp}{\vdash \uparrow (\mathbf{1} \oplus \mathbf{1}), \phi^-(\Gamma)^\perp}} \qquad \frac{\dfrac{}{\vdash \mathbf{1}, \phi^+(\Gamma)^\perp}}{\vdash \uparrow \mathbf{1}, \phi^+(\Gamma)^\perp}$$

- Case:

$$\frac{\dfrac{\vdash \uparrow \phi^+(A), \phi^+(\Gamma)^\perp}{\vdash \uparrow \phi^+(A), \perp, \phi^+(\Gamma)^\perp} \qquad \dfrac{\vdash \uparrow \phi^+(A), \phi^+(\Gamma)^\perp}{\vdash \uparrow \phi^+(A), \perp, \phi^+(\Gamma)^\perp}}{\vdash \uparrow \phi^+(A), \perp \& \perp, \phi^+(\Gamma)^\perp}$$

Using this translation, call-by-value reduction of lambda terms can be simulated using cut elimination of LLP [53].

From CBV to WS

The composition of the above to translations give an interpretation $i \circ \phi^+$ of CBV inside WS1. Here we discuss the result of this translation.

First, we note that $i(\phi^+(2)) = i(\mathbf{1} \oplus \mathbf{1}) = \{\mathbf{1}, \mathbf{1}\}$ which is a pair of formulas, unlike in CBN which used a singleton formulation. This is because Booleans are ground types, and in call-by-value a term of ground type is an element of a set rather than a computation. Nonetheless, a term of type 2 is interpreted as a LLP proof of $\vdash \uparrow (\mathbf{1} \oplus \mathbf{1})$ which corresponds to a WS1 proof of $\vdash \perp, ((\top \oslash \mathbf{1}) \oplus (\top \oslash \mathbf{1})$ i.e. $\vdash \perp, \top \oplus \top$.

The type $2 \to 2$ is translated to the LLP formula $!((\perp \& \perp) \parr \uparrow (\mathbf{1} \oplus \mathbf{1})$. Since $i(\uparrow (\mathbf{1} \oplus \mathbf{1})) = \{((\top \oslash \mathbf{1}) \oplus (\top \oslash \mathbf{1}))\} \cong \{\top \oplus \top\}$, we have $i((\perp \& \perp) \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) = \{\uparrow (\mathbf{0} \parr (\top \oplus \top)), \uparrow (\mathbf{0} \parr (\top \oplus \top))\} \cong \{\top \oplus \top, \top \oplus \top\}$. Finally, $i(!((\perp \& \perp) \parr \uparrow (\mathbf{1} \oplus \mathbf{1}))) = \{!((\perp \lhd (\top \oplus \top)) \& (\perp \lhd (\top \oplus \top)))\}$.

Thus a term of type $2 \to 2$ is translated to a LLP proof of $\vdash \uparrow !((\perp \& \perp) \parr ?(\mathbf{1} \oplus \mathbf{1})$, which is mapped to a WS1 proof of $\vdash \perp, \top \oslash !((\perp \lhd (\top \oplus \top)) \& (\perp \lhd (\top \oplus \top)))$. It is worth comparing a dialogue in this game with a dialogue in the CBN equivalent $!\mathbf{B} \multimap \mathbf{B}$.

- In the CBN version, Opponent asks for the output Boolean (output move in $\mathbf{B}$). Player then may ask for the input Boolean multiple times and Opponent is under no obligation to give the same one each time (moves in input $!\mathbf{B}$). Finally, Player can produce the output Boolean (in $\mathbf{B}$).
- In the CBV version, Opponent indicates that he wishes to call the function (opening $\perp$ move). Player may then interact with the environment, perhaps as he evaluates earlier arguments. Once completed, Player plays $\top$.

Opponent can then (repeatedly) use the resulting function. He does this by giving the input Boolean (which is just a single move — any term of type 2 must already have been evaluated previously). Player can then (after further interaction with the environment) give the output Boolean.

If a, b and c are terms of type 2 (which may nonetheless interact with the environment), the application rule ensures that in the term $(\lambda xy.c)ab$ these interactions occur in the following order: a, then b, then c. This is reflected in the translation of the application rule to a WS proof, which we demonstrate here in the case that the argument (and everything in the environment) is a singleton family. For brevity we write T for $i(\phi^+(T))$ and S for $i(\phi^+(S))$.

$$\mathsf{P}_{\mathsf{cut}}\ \dfrac{\dfrac{\vdash \bot, \top \oslash !(\bot \lhd (S^{\bot} \parr (\top \oslash T)))) \qquad q}{\vdash \bot, \top \oslash T, i(\phi^+(\Gamma)^{\bot}), i(\phi^+(\Gamma)^{\bot})}}{\vdash \bot, \top \oslash T, i(\phi^+(\Gamma)^{\bot})}$$

where q is:

$$\mathsf{P}_{\mathsf{cut}}\ \dfrac{\vdash \bot, \top \oslash S, i(\phi^+(\Gamma)^{\bot}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash S, S^{\bot} \quad \vdash \bot \lhd T^{\bot}, \top \oslash T}{\vdash S \otimes (\bot \lhd T^{\bot}), S^{\bot}, \top \oslash T}\mathsf{P}_{\mathsf{mul}\otimes}}{\vdash \top, (S \otimes (\bot \lhd T^{\bot})), S^{\bot}, \top \oslash T}}{\vdash \top \oslash (S \otimes (\bot \lhd T^{\bot})), S^{\bot}, \top \oslash T}}{\vdash ?(\top \oslash (S \otimes (\bot \lhd T^{\bot}))), S^{\bot}, \top \oslash T}}{\vdash \bot, S^{\bot} \parr ?(\top \oslash (S \otimes (\bot \lhd T^{\bot}))) \parr \top \oslash T}}{\vdash \bot, S^{\bot}, ?(\top \oslash (S \otimes (\bot \lhd T^{\bot}))), \top \oslash T}}{\vdash \bot \lhd S^{\bot}, ?(\top \oslash (S \otimes (\bot \lhd T^{\bot}))), \top \oslash T}}{\vdash \bot, ?(\top \oslash (S \otimes (\bot \lhd T^{\bot}))), \top \oslash T, i(\phi^+(\Gamma)^{\bot})}}{\vdash \bot \lhd ?(\top \oslash (S \otimes (\bot \lhd T^{\bot}))), \top \oslash T, i(\phi^+(\Gamma)^{\bot})}$$

5.2 Call-by-value Finitary Imperative Language

In the previous section, we gave an embedding of finitary call-by-name and call-by-value lambda calculi inside WS1. In the next sections we describe how this embedding can be extended to more expressive languages.

We will first show how the call-by-value embedding via LLP can be extended with imperative features. In particular, we consider ground state, coroutining, and an encapsulation operator reflecting the $\mathsf{P}_{\mathsf{ana}}$ rule.

5.2.1 A Call-by-value Finitary Imperative Language

We extend CBV λ_{fin} with some imperative features. Firstly we note that in a call-by-value setting, imperative sequencing $a; b$ can be simulated by $(\lambda x.b)a$ where x is a fresh variable. Similarly, $\mathsf{let}(x, e, M)$ (let x be e in M) can be defined as $(\lambda x.M)e$, and e will be evaluated before M.

- **Storage Cell** — A Boolean storage cell can be represented by the product of its "read" and "write" methods, of type $(1 \to 2) \times (2 \to 1)$. We introduce a new constant $\mathsf{cell} : 2 \to (1 \to 2) \times (2 \to 1)$ which generates a new storage cell with a given starting value.
- **Coroutines** — We can introduce a deterministic form of multithreading by defining a coroutining constant $\mathsf{co} : ((1 \to 1) \to 1) \to ((1 \to 1) \to 1) \to 1$. A term of type $(1 \to 1) \to 1$ can be seen as a command which may call its argument (a command, of type $1 \to 1$) multiple times. The idea is that in $\mathsf{co}\ a\ b$ we run a and b in an interleaved manner, where control starts in a and is passed to b when (and if) a calls its argument. When b calls its argument, control is passed back to a, and so on. Whenever either a or b terminates, then $\mathsf{co}\ a\ b$ terminates.
- **Encaps** — We can represent a mild form of higher-order state using a constant $\mathsf{encaps} : (s \to (o \times s)) \to s \to 1 \to o$. The idea is that $t = \mathsf{encaps}\ f\ s_0 : 1 \to o$ represents a stream of output values, computed from an internal state. The internal state starts as s_0 and each time $t()$ is invoked f is applied to the internal state, which determines the result of the current $t()$ call and the new internal state. Note that s and o can be arbitrary types. An ML implementation can be given using general reference cells:

  ```
  (* val encaps : ('s -> 'o * 's) -> 's -> unit -> 'o *)
  let encaps f i =
    let s = ref i in
    fun (_ : unit) ->
      let (o,ns) = f !s in
      s := ns; o;;
  ```

  The encaps operation is the program interpretation of the $\mathsf{P}_{\mathsf{ana}}$ rule.

  For simplicity of our WS1 translation, we will assume that any occurrence of 2 in s or o occurs under an arrow: we require that they are not "value types" and so s and o are translated to singleton families in WS1.

We will call the CBV λ_{fin} with these constants *TotLangV*.

Remark The encaps operator appears in [58]. It also corresponds to the thread operator in [75], which is used for interpreting the internal state of imperative objects in a Curien-Lamarche games model. The type of thread corresponds to $(p \times s \to o \times s) \to s \to (p \to o)$ where p is a ground type. In our setting p is forced to be 1, but the more general form can be be emulated using a reference cell written by the environment before each call. Dually, ground reference cells can be defined using this more general form of encaps.

We can make a similar remark for coroutines. In [48], the two coroutines can pass *ground values* to each other when they also pass control. This message-passing can be simulated in our setting using shared ground variables; and ground variables can be simulated using message-passing coroutines.

We can use encaps and cell to define a stack of Booleans, with push and pop operations, following Figure 3-4. In using encaps, the state type is $1 \to 2$ and the output type is $(2 \to 1) \times (1 \to 2)$. We use ML concrete syntax (`ref` corresponds to cell; assignment and dereferencing to projections; and `if-then-else` to case).

```
(* val newstack : unit -> (bool -> unit) * (unit -> bool) *)
let newstack = fun (_:unit) ->
  let f = fun (pop : unit -> bool) ->
    let pushed = ref false in
    let valu = ref false in
    let push b = pushed := true; valu := b in
    let newpop() = if !pushed then (pushed := false; !valu) else pop() in
    ((push,pop),newpop) in
  let i = fun (_:unit) -> false in
  let stack = encaps f i in
  let push b = fst (stack()) b in
  let pop () = snd (stack()) () in
  (push,pop);;
```

Testing this in an ML interpreter gives the expected result.

Operational Semantics

We next give operational semantics of TotLangV, which we intend our WS1 embedding to respect. We use a small-step style, adding two state components to the terms. First, a ground store G, which is a partial mapping from locations to Booleans. Second, a stream store ST, which is a partial mapping from locations to pairs of terms (v, t) where $v : s \to o \times s$ and $t : s$. The operation fr picks a fresh location. We introduce some auxiliary constants:

- If l is a G-location, we add $\delta_l : 1 \to 2$ for dereferencing and and $\alpha_l : 2 \to 1$ for assignment
- If r is a ST-location with types s and o, we add $\nu_r : 1 \to o$ requesting a new element of the stream r, and $\rho_r : o \times s \to o$ representing a term which acts like π_1 but also updates the second component of r.

We next define the class of values. Here M ranges over terms, V over values and pk over *partially applied* constants — terms $kM_1 \dots M_n$ where k is a constant and $n < \mathsf{ar}(\mathtt{k})$. The arity of encaps is 2.

$$V \quad := \quad x \quad | \quad pk \quad | \quad () \quad | \quad \mathtt{tt} \quad | \quad \mathtt{ff} \quad | \quad \langle V, V \rangle \quad | \quad \lambda x.M$$

Evaluation contexts are given by

$$E[\_] :=$$
$$\_ \mid E[\_]M \mid VE[\_] \mid \langle E[\_], M\rangle \mid \langle V, E[\_]\rangle \mid \mathsf{case}(E[\_], M, M) \mid \mathsf{co}(\lambda x.E[\_], M)$$

Every term M can be written in the form $E[N]$ for some evaluation context E and term N. The idea is that the first step of evaluating M consists of evaluating N inside the context E.

We give the small-step operational semantics for TotLangV in Figure 5-2. The reduction relation is between tuples (G, ST, t) where G is a ground store, ST is a stream store and t is a term. The metavariable F_x ranges over evaluation contexts that do not bind x. If either of the state components is preserved, we omit it.

5.2.2 Embedding into WS1

We next embed TotLangV into $\mathsf{WS1}$. For the CBV λ_{fin} fragment, we use the embedding via LLP defined above, and so we only need to embed the three imperative constants. The embedding of cell follows the game semantics given in [8]; the embedding of co follows the game semantics given in [51]; the embedding of encaps uses the anamorphism rule of $\mathsf{WS1}$. Recall the notation $\uparrow P = \bot \lhd P$ and $\downarrow N = \top \oslash N$ on $\mathsf{WS1}$ formulas.

Boolean Cell

$\phi^+(2 \to (1 \to 2) \times (2 \to 1))$ is the LLP formula

$$C = !((\bot \& \bot) \parr \uparrow (!(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1}))$$

and to be compatible with our λ_{fin} embedding we must give a LLP proof of $\vdash \uparrow C$ or a $\mathsf{WS1}$ proof of $\vdash \bot, \top \oslash i(C)$. We will in fact use a combination of these approaches,

Figure 5-2: Operational Semantics of TotLangV

$$
\begin{array}{rcl}
E[(\lambda x.M)\ V] & \to & E[M\{V/x\}] \\
E[\mathsf{case}(\mathtt{tt}, M, N)] & \to & E[M] \\
E[\mathsf{case}(\mathtt{ff}, M, N)] & \to & E[N] \\
E[\pi_1(\langle V_1, V_2\rangle)] & \to & E[V_1] \\
E[\pi_2(\langle V_1, V_2\rangle)] & \to & E[V_2] \\
E[\mathsf{cell}\ V_b], G & \to & E[\langle \lambda\_.\delta_l\ (), \lambda x.\alpha_l\ x\rangle], G[l \mapsto V_b] \\
 & & \text{where } l = \mathsf{fr}(G) \\
E[\delta_l\ ()], G[l \mapsto b] & \to & E[b], G[l \mapsto b] \\
E[\alpha_l\ V_b], G[l \mapsto \_] & \to & E[()], \sigma[l \mapsto V_b] \\
E[\mathsf{co}\ (\lambda x.F_x[x()])\ M] & \to & E[\mathsf{co}\ M\ (\lambda x.F_x[()])] \\
E[\mathsf{co}\ (\lambda x.())\ M] & \to & E[()] \\
E[\mathsf{encaps}\ V_f\ V_s], ST & \to & E[\nu_r], ST[r \mapsto (V_f, V_s)] \\
 & & \text{where } r = \mathsf{fr}(ST) \\
E[\nu_r\ ()], ST[r \mapsto (f, s)] & \to & E[\rho_r\ (fs)], ST[r \mapsto (f, s)] \\
E[\rho_r\ \langle V_o, V_s\rangle], ST[r \mapsto (f, \_)] & \to & E[V_o], ST[r \mapsto (f, V_s)]
\end{array}
$$

translating a fragment of LLP proof that massages the types, together with a WS1 representing the history-sensitive behaviour. The former is as follows:

$$
\dfrac{\dfrac{\dfrac{\dfrac{q_1 \vdash !(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1})}{\vdash \bot, !(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1})} \quad \dfrac{q_2 \vdash !(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1})}{\vdash \bot, !(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1})}}{\vdash \bot \& \bot, !(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1})}}{\vdash \bot \& \bot, \uparrow (!(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1})}}{\vdash (\bot \& \bot) \parr \uparrow (!(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1})}}{\vdash !((\bot \& \bot) \parr \uparrow (!(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1}))}
$$

Using the WS1 translation of the above as the start (bottom) of our proof, we require proofs $q_1, q_2 \vdash \bot, \top \oslash i(!(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1}))$ representing the history-sensitive behaviour of the Boolean cell, for each possible starting value. Using $\mathsf{P}^{+}_{\bot}(\mathsf{P}_{\oslash}(\mathsf{P}^{-}_{\top}(-)))$ it is sufficient to give proofs of $\vdash i(!(\bot \parr \uparrow (\mathbf{1} \oplus \mathbf{1})) \otimes !((\bot \& \bot) \parr \uparrow \mathbf{1}))$, which is

$$
\vdash !(\bot \lhd (\mathbf{0} \parr (\top \oslash \mathbf{1} \oplus \top \oslash \mathbf{1}))) \otimes !((\bot \lhd (\mathbf{0} \parr (\top \oslash \mathbf{1}))) \& (\bot \lhd (\mathbf{0} \parr (\top \oslash \mathbf{1}))))
$$

which is isomorphic to $!(\bot \lhd (\top \oplus \top)) \otimes !((\bot \lhd \top) \& (\bot \lhd \top)) \cong !(\mathbf{B} \& \mathbf{Bi})$ with isomorphisms definable in WS1. We can obtain our required proof by composing this isomorphism with the Boolean cell proof given in Section 3.4.3.

Coroutines

By using the translation of abstraction, it is sufficient to give a proof of $\vdash \bot, \top \oslash i \circ \phi^+(1), i \circ \phi^+((1 \to 1) \to 1)^\bot, i \circ \phi^+((1 \to 1) \to 1)^\bot$ i.e. $\vdash \bot, \top \oslash \mathbf{1}, ? \downarrow (! \uparrow (\mathbf{0} \parr \downarrow \mathbf{1}) \otimes ! \uparrow \mathbf{0}), ? \downarrow (! \uparrow (\mathbf{0} \parr \downarrow \mathbf{1}) \otimes ! \uparrow \mathbf{0})$ which is isomorphic to $\vdash \bot, \top, ?(\top \oslash (!(\bot \lhd \top) \otimes \bot), ?(\top \oslash (!(\bot \lhd \top) \otimes \bot)$. This is in turn isomorphic to $\vdash \Sigma, ?(\Sigma^\bot \oslash !\Sigma), ?(\Sigma^\bot \oslash !\Sigma)$ where $\Sigma = \bot \lhd \top$, with isomorphisms definable in WS1. Thus it suffices to give a proof cocomp of $\vdash \Sigma, ?(\Sigma^\bot \oslash !\Sigma), ?(\Sigma^\bot \oslash !\Sigma)$. We first give a proof o of $(!\Sigma \multimap \bot) \multimap !\Sigma$.

$$\mathsf{P_{ana}}\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P_{id}}\ \dfrac{}{\vdash \bot \lhd ?(\top \oslash \bot), \top \oslash !(\bot \lhd \top)}}{\vdash \top, \bot \lhd ?(\top \oslash \bot), \top \oslash !(\bot \lhd \top)}}{\vdash \top \oslash (\bot \lhd ?(\top \oslash \bot)), \top \oslash !(\bot \lhd \top)}}{\vdash \bot, \top \oslash !(\bot \lhd \top), \top \oslash (\bot \lhd ?(\top \oslash \bot))}}{\vdash \bot \lhd \top, !(\bot \lhd \top), \top \oslash (\bot \lhd ?(\top \oslash \bot))}}{\vdash !(\bot \lhd \top), \top \oslash (\bot \lhd ?(\top \oslash \bot))}}{\vdash \top, !(\bot \lhd \top), \top \oslash (\bot \lhd ?(\top \oslash \bot))}}{\vdash \top \oslash !(\bot \lhd \top), \top \oslash (\bot \lhd ?(\top \oslash \bot))}}{\vdash \bot, \top \oslash (\bot \lhd ?(\top \oslash \bot)), \top \oslash !(\bot \lhd \top)}}{\vdash \bot, \top, \bot \lhd ?(\top \oslash \bot), \top \oslash !(\bot \lhd \top)}}{\vdash \bot \lhd \top, \bot \lhd ?(\top \oslash \bot), \top \oslash !(\bot \lhd \top)}}{\vdash !(\bot \lhd \top), \top \oslash !(\bot \lhd \top)}$$

Remark The above defines an isomorphism in $\mathcal{G}$, but not in $\mathcal{W}$ as its inverse is not winning.

We next define a proof $o' \vdash (!\Sigma \multimap \Sigma) \multimap \bot \multimap !\Sigma$, which connects the output move of the first argument to the Player-move in the second argument.

$$\mathsf{P_{cut}}\ \dfrac{o \vdash !\Sigma, \top \oslash !\Sigma \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P_{mul}}\otimes\ \dfrac{\dfrac{}{\vdash \bot, \top} \quad \dfrac{}{\vdash !\Sigma, ?\Sigma^\bot}}{\vdash \bot \otimes !\Sigma, ?\Sigma^\bot, \top}}{\vdash \top, \bot, !\Sigma, ?\Sigma^\bot, \top}}{\vdash (\top \oslash \bot) \oslash !\Sigma, ?\Sigma^\bot, \top}}{\vdash \bot, ?\Sigma^\bot, \top, (\top \oslash \bot) \oslash !\Sigma}}{\vdash \bot \lhd ?\Sigma^\bot, \top, (\top \oslash \bot) \oslash !\Sigma}}{\vdash !\Sigma, \top, (\top \oslash \bot) \oslash !\Sigma}}{\vdash !\Sigma \lhd \top, \Sigma^\bot \oslash !\Sigma}$$

We can then define cocomp.

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$\vdash \top$}
\UnaryInfC{$\vdash \bot, \top$}
\AxiomC{}
\UnaryInfC{$\vdash \top$}
\UnaryInfC{$\vdash \bot, \top$}
\BinaryInfC{$\vdash \bot \otimes \bot, \top$}
\UnaryInfC{$\vdash \top, \bot, \bot, \top$}
\UnaryInfC{$\vdash (\top \oslash \bot) \oslash \bot, \top$}
\UnaryInfC{$\vdash \bot, \top, (\top \oslash \bot) \oslash \bot$}
\UnaryInfC{$\vdash \Sigma, \Sigma^{\perp} \oslash \bot$}
\AxiomC{}
\LeftLabel{$\mathsf{P_{id}}$}
\UnaryInfC{$\vdash \Sigma, \Sigma^{\perp}$}
\AxiomC{}
\LeftLabel{$\mathsf{P_{id}}$}
\UnaryInfC{$\vdash !\Sigma, ?\Sigma^{\perp}$}
\AxiomC{}
\LeftLabel{$\mathsf{P_{id}}$}
\UnaryInfC{$\vdash \bot, \top$}
\LeftLabel{$\mathsf{P_{\multimap}}$}
\BinaryInfC{$\vdash !\Sigma, ?\Sigma^{\perp} \oslash \bot, \top$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash !\Sigma, \top, ?\Sigma^{\perp} \oslash \bot$}
\LeftLabel{$\mathsf{P_{\multimap}}$}
\BinaryInfC{$\vdash \Sigma, \Sigma^{\perp} \oslash !\Sigma, \top, ?\Sigma^{\perp} \oslash \bot$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash \Sigma, \top, \Sigma^{\perp} \oslash !\Sigma, ?\Sigma^{\perp} \oslash \bot$}
\UnaryInfC{$\vdash \Sigma \lhd \top, \Sigma^{\perp} \oslash !\Sigma, ?\Sigma^{\perp} \oslash \bot$}
\LeftLabel{$\mathsf{P_{cut}}$}
\BinaryInfC{$\vdash \Sigma, \Sigma^{\perp} \oslash !\Sigma, ?\Sigma^{\perp} \oslash \bot$}
\AxiomC{o'}
\LeftLabel{$\mathsf{P_{cut}}$}
\BinaryInfC{$\vdash \Sigma, \Sigma^{\perp} \oslash !\Sigma, \Sigma^{\perp} \oslash !\Sigma$}
\end{prooftree}
$$

The semantics of cocomp agrees with the treatment given in [48, 51].

Encaps

Let $\{S\} = i \circ \phi^{+}(s)$ and $\{O\} = i \circ \phi^{+}(o)$. By using the translation of abstraction, it is sufficient to give a proof of $\vdash \bot, \top \oslash i \circ \phi^{+}(1 \to o), S^{\perp}, i \circ \phi^{+}(s \to (o \times s))^{\perp}$ i.e. $\vdash \bot, \top \oslash !(\mathbf{0} \parr \uparrow O), S^{\perp}, ? \downarrow (S^{\perp} \otimes \uparrow (O^{\perp} \parr S^{\perp}))$. We do so:

$$
\begin{prooftree}
\AxiomC{a}
\AxiomC{}
\LeftLabel{$\mathsf{P_{id}}$}
\UnaryInfC{$\vdash ! \uparrow (S^{\perp} \parr \downarrow (O \otimes S)), ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp}))$}
\AxiomC{b}
\LeftLabel{$\mathsf{P_{mul}}$}
\BinaryInfC{$\vdash \uparrow (\downarrow O), ! \uparrow (S^{\perp} \parr \downarrow (O \otimes S)), S, ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})), ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})), S^{\perp}$}
\LeftLabel{$\mathsf{P^{?}_{con}}$}
\UnaryInfC{$\vdash \uparrow (\downarrow O), ! \uparrow (S^{\perp} \parr \downarrow (O \otimes S)), S, ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})), S^{\perp}$}
\UnaryInfC{$\vdash \uparrow (\downarrow O), ! \uparrow (S^{\perp} \parr \downarrow (O \otimes S)) \otimes S, ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})) \parr S^{\perp}$}
\LeftLabel{$\mathsf{P_{ana}}$}
\UnaryInfC{$\vdash ! \uparrow (\downarrow O), ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})) \parr S^{\perp}$}
\LeftLabel{$\mathsf{P_{cut}}$}
\BinaryInfC{$\vdash ! \uparrow (\mathbf{0} \parr \downarrow O), ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})) \parr S^{\perp}$}
\LeftLabel{$\mathsf{P^{\top}_{\parr}}$}
\UnaryInfC{$\vdash ! \uparrow (\mathbf{0} \parr \downarrow O)), ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})), S^{\perp}$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash ! \uparrow (\mathbf{0} \parr \downarrow O)), S^{\perp}, ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp}))$}
\UnaryInfC{$\vdash \top, ! \uparrow (\mathbf{0} \parr \downarrow O)), S^{\perp}, ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp}))$}
\UnaryInfC{$\vdash \bot, \top \oslash ! \uparrow (\mathbf{0} \parr \downarrow O)), S^{\perp}, ? \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp}))$}
\end{prooftree}
$$

where a is the evident isomorphism $\vdash ! \uparrow (\mathbf{0} \parr \downarrow O), ? \downarrow \uparrow O^{\perp}$ and b is:

$$
\mathsf{P^{?}_{der}}\,\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P^{+}_{sym}}\,\dfrac{\mathsf{P_{mul\otimes}}\,\dfrac{\mathsf{P_{id}}\,\dfrac{}{\vdash S, S^{\perp}} \qquad \dfrac{\dfrac{\dfrac{\mathsf{P}^{\top}_{\parr}\,\dfrac{\dfrac{\mathsf{P_{mul}}\,\dfrac{\mathsf{P_{id}}\,\dfrac{}{\vdash O, O^{\perp}} \qquad \mathsf{P_{id}}\,\dfrac{}{\vdash S, S^{\perp}}}{\vdash O, S, O^{\perp}, S^{\perp}}}{\vdash O \oslash S, O^{\perp}, S^{\perp}}}{\vdash O \oslash S, O^{\perp} \parr S^{\perp}}}{\vdash \top, O \oslash S, O^{\perp} \parr S^{\perp}}}{\vdash O^{\perp} \parr S^{\perp} \parr (\downarrow O \oslash S)}}{\vdash \bot, O^{\perp} \parr S^{\perp}, \downarrow O \oslash S}}{\vdash S \otimes \uparrow (O^{\perp} \parr S^{\perp}), \downarrow O \oslash S, S^{\perp}}}{\vdash S \otimes \uparrow (O^{\perp} \parr S^{\perp}), S^{\perp}, \downarrow O \oslash S}}{\vdash \top, S \otimes \uparrow (O^{\perp} \parr S^{\perp}), \downarrow O \oslash S, S^{\perp}}}{\vdash \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})), \downarrow O \oslash S, S^{\perp}}}{\vdash \bot, (\downarrow O \oslash S) \parr \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})) \parr S^{\perp}}}{\vdash \uparrow (\downarrow O), S, \downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})), S^{\perp}}}{\vdash \uparrow (\downarrow O), S, ?\downarrow (S \otimes \uparrow (O^{\perp} \parr S^{\perp})), S^{\perp}}
$$

5.3 The Logic WSN

In the next section, we will embed a (call-by-name) language with a base type of natural numbers into our logic. In order to do this, we must be able to represent the infinitely wide game (with a possible move for each natural number) as a formula. This is not possible in WS1. However, we can add it without breaking any of the results achieved in previous chapters.

5.3.1 The Logic WSN

Formulas

We extend the formulas of WS1 as follows:

$$
\begin{aligned}
M, N &:= \quad \ldots \mid \omega \\
P, Q &:= \quad \ldots \mid \overline{\omega}
\end{aligned}
$$

- The formula ω denotes the game where Opponent has a move for each natural number, with no responses (so ω represents the countable product of $\bot$ — we have $\omega \cong \bot \& \omega$).
- The formula $\overline{\omega}$ denotes the game where Player has a move for each natural number, with no Opponent-responses (so $\overline{\omega}$ represents the countable sum of $\top$ — $\overline{\omega} \cong \top \oplus \overline{\omega}$).

We can encode the game of natural numbers (see Section 2.1.1) as $\bot \lhd \overline{\omega}$.

Figure 5-3: Proof rules for WSN — extends Figure 4-1

Core rules:

$$\mathsf{P}_\omega \frac{X;\Theta \vdash \bot, \Gamma \qquad X;\Theta \vdash \omega, \Gamma}{X;\Theta \vdash \omega, \Gamma} \qquad \mathsf{P}^n_{\overline{\omega}} \frac{X;\Theta \vdash \top, \Gamma}{X;\Theta \vdash \overline{\omega}, \Gamma}$$

Other rules:

$$\mathsf{P}_0 \frac{}{X;\Theta \vdash \overline{\omega}} \qquad \mathsf{P}_{\mathsf{suc}} \frac{}{X;\Theta \vdash \omega, \overline{\omega}}$$

$$\mathsf{P}_{\mathsf{ind}} \frac{X;\Theta \vdash P \qquad X;\Theta \vdash P^\bot, P}{X;\Theta \vdash \omega, P} \qquad \mathsf{P}^{\mathsf{lfe}}_\omega \frac{X;\Theta \vdash \omega, P \oslash N, \Gamma}{X;\Theta \vdash \omega, P, N, \Gamma}$$

Proof Rules

The proof rules of WSN are given in Figure 5-3. Note that there is a rule $\mathsf{P}^n_{\overline{\omega}}$ for each $n \in \mathbb{N}$. We informally describe each rule:

- Given strategies σ and τ, $\mathsf{P}_\omega(\sigma, \tau)$ plays as σ if Opponents first move is 0, or as $\tau(n)$ if Opponents first move is $n+1$.
- $\mathsf{P}^n_{\overline{\omega}}$ plays the natural number n as its first move, and then plays as its premise.
- The proof P_0 denotes the strategy where Player plays 0, and play ends.
- The proof $\mathsf{P}_{\mathsf{suc}}$ denotes the strategy where Opponent plays natural n and Player responds by playing $n+1$.
- $\mathsf{P}_{\mathsf{ind}}$ is an inductive rule. The first premise gives Player's response in the case that Opponent plays 0, and the second premise gives the 'inductive step'.
- $\mathsf{P}^{\mathsf{lfe}}_\omega$ is interpreted by a game isomorphism.

5.3.2 Semantics of WSN

To model the new formulas and proof rules, we simply require that the object $\bot$ has a countable product $\bot^\omega$. We then set $[\![\omega]\!] = [\![\overline{\omega}]\!] = \bot^\omega$ (the constant functor).

Proposition 5.3.1 *Let $\mathcal{C}$ be a Cartesian category, and A^ω a countable product of A. Then $X \mapsto A \times X$ has a final coalgebra, whose carrier is A^ω.*

Proof We write π_n with $n \in \mathbb{N}$ for the projections $A^\omega \to A$ and $\langle f(i) \rangle_{i \in \mathbb{N}}$ for tupling (with usual π_1 and π_2 for binary projections). The morphism $\alpha : A^\omega \to A \times A^\omega$ is given by $\langle \pi_0, \langle \pi_{n+1} \rangle_{n \in \mathbb{N}} \rangle$. Note that α is an isomorphism, with $\alpha^{-1} = \langle g_n \rangle_{n \in \mathbb{N}}$ with $g_0 = \pi_1$ and $g_{n+1} = \pi_n \circ \pi_2$.

Figure 5-4: Semantics of WSN — extends Figure 4-3

$$\mathsf{P}_\omega \frac{\sigma : [\![X; \Theta \vdash \bot, \Gamma]\!] \qquad \tau : [\![X; \Theta \vdash \omega, \Gamma]\!]}{[\![\Gamma]\!]^-(\alpha^{-1}) \circ \mathsf{dist}^{-1}_{\Gamma,-} \circ \langle \sigma, \tau \rangle : [\![X; \Theta \vdash \omega, \Gamma]\!]} \quad \mathsf{P}^n_{\overline{\omega}} \frac{\sigma : [\![X; \Theta \vdash \top, \Gamma]\!]}{\sigma \circ [\![\Gamma]\!]^+(\pi_n) : [\![X; \Theta \vdash \overline{\omega}, \Gamma]\!]}$$

$$\mathsf{P}_0 \frac{}{\pi_1 \circ \alpha : [\![X; \Theta \vdash \overline{\omega}]\!]} \quad \mathsf{P}_{\mathsf{suc}} \frac{}{\Lambda_I(\pi_2 \circ \alpha) : [\![X; \Theta \vdash \omega, \overline{\omega}]\!]}$$

$$\mathsf{P}_{\mathsf{ind}} \frac{\sigma : [\![X; \Theta \vdash P]\!] \qquad \tau : [\![X; \Theta \vdash P^\perp, P]\!]}{\Lambda_I(\mathbb{(} \langle \sigma, \Lambda_I^{-1}(\tau) \rangle \mathbb{)}) : [\![X; \Theta \vdash \omega, P]\!]}$$

$$\mathsf{P}^{\mathsf{lfe}}_\omega \frac{\sigma : [\![X; \Theta \vdash \omega, P \oslash N, \Gamma]\!]}{[\![\Gamma]\!]^-(\mathsf{lfe}^{-1}) \circ \sigma : [\![X; \Theta \vdash \omega, P, N, \Gamma]\!]}$$

Given $f : B \to A \times B$ we define $\mathbb{(} f \mathbb{)} : B \to A^\omega$ as $\langle f_n \rangle_{n \in \mathbb{N}}$ where $f_0 = \pi_1 \circ f$ and $f_{n+1} = f_n \circ \pi_2 \circ f$. We need to show that $\mathbb{(} f \mathbb{)}$ is the unique map such that $\alpha \circ \mathbb{(} f \mathbb{)} = (\mathsf{id} \times \mathbb{(} \alpha \mathbb{)}) \circ f$.

To show that the equation holds, note that $\alpha \circ \mathbb{(} f \mathbb{)} = \langle \pi_0, \langle \pi_{n+1} \rangle_{n \in \mathbb{N}} \rangle \circ \langle f_n \rangle = \langle \pi_0 \circ \langle f_n \rangle, \langle \pi_{n+1} \rangle_{n \in \mathbb{N}} \langle f_n \rangle \rangle = \langle f_0, \langle f_{n+1} \rangle_{n \in \mathbb{N}} \rangle$. Similarly, $(\mathsf{id} \times \mathbb{(} \alpha \mathbb{)}) \circ f = (\mathsf{id} \times \langle f_n \rangle_{n \in \mathbb{N}}) \circ f = (\mathsf{id} \times \langle f_n \rangle_{n \in \mathbb{N}}) \circ \langle \pi_1, \pi_2 \rangle \circ f = (\mathsf{id} \times \langle f_n \rangle_{n \in \mathbb{N}}) \circ \langle \pi_1 \circ f, \pi_2 \circ f \rangle = \langle \pi_1 \circ f, \langle f_n \rangle_{n \in \mathbb{N}} \circ \pi_2 \circ f \rangle = \langle f_0, \langle f_n \circ \pi_2 \circ f \rangle_{n \in \mathbb{N}} \rangle = \langle f_0, \langle f_{n+1} \rangle_{n \in \mathbb{N}} \rangle$.

For uniqueness, suppose $\alpha \circ g = (\mathsf{id} \times g) \circ f$. We show that for each i, $\pi_i \circ g = f_i$, by induction on i. For $i = 0$ we have $\pi_0 \circ g = \pi_0 \circ \alpha^{-1} \circ (\mathsf{id} \times g) \circ f = \pi_1 \circ (\mathsf{id} \times g) \circ f = \mathsf{id} \circ \pi_1 \circ f = \pi_1 \circ f = f_0$. For $i = j + 1$ we have $\pi_i \circ g = \pi_i \circ \alpha^{-1} \circ (\mathsf{id} \times g) \circ f = \pi_j \circ \pi_2 \circ (\mathsf{id} \times g) \circ f = \pi_j \circ g \circ \pi_2 \circ f = f_j \circ \pi_2 \circ f = f_{j+1}$ as required. ∎

Proposition 5.3.2 *In any WS-category with countable products, There is an isomorphism* $\mathsf{lfe} : (B \multimap \bot^\omega) \oslash A \Rightarrow (A \multimap B) \multimap \bot^\omega$.

Proof Using countable distributivity and linear functional extensionality of $\bot$, we note that $(B \multimap \bot^\omega) \oslash A \cong ((B \multimap \bot) \oslash A)^\omega \cong ((A \multimap B) \multimap \bot)^\omega \cong (A \multimap B) \multimap \bot^\omega$. ∎

We can use the above to interpret the proof rules of WSN. Note that the new rules do not vary the $X; \Theta$ context, and both $\mathcal{W}^{\mathcal{M}^\Theta_X}$ and $\mathcal{G}^{\mathcal{M}^\Theta_X}$ have countable products (lifted pointwise). We give semantics to WSN in Figure 5-4.

5.3.3 Full Completeness

An analytic proof in WSN is a proof using only the core rules, satisfying the leanness restrictions in Section 4.3. If σ is a uniform winning strategy on $[\![X; \Theta \vdash \Gamma]\!]$, we say σ

is *finitary* if $\sigma_{(L,v)}$ is finite whenever $|L|$ is. By using a countable version of π-atomicity, we can show that:

Proposition 5.3.3 *In WSN, each finitary uniform winning strategy on* $[\![X; \Theta \vdash \Gamma]\!]$ *is the denotation of a unique analytic proof.*

Proof We extend the results of previous chapters. In particular, any uniform winning strategy on $[\![\overline{\omega}, \Gamma]\!]$ corresponds to a particular projection π_n and a uniform winning strategy on $[\![\top, \Gamma]\!]$. For the P_ω rule, we note that if $\sigma : [\![\omega, \Gamma]\!]$ is finitary then it can respond to only finitely many inputs, and so isn't total, and so this case never arises.

The procedure terminates using the same inductive measure: any finitary strategy on a WS1-type must be bounded. To see this, note that the maximum play size over all $\sigma_{(L,v)}$ is the same as the maximum play size in $\sigma_{(L,v)}$ for any particular (L, v) satisfying all positive atoms. By picking such an $|L|$ that is finite, we see that $\sigma_{(L,v)}$ is finite, hence bounded. And so σ is bounded. ∎

The comment above regarding P_ω indicates that the above proposition is of limited use, as formulas involving ω will not typically be inhabited by any finitary uniform winning strategy. However, as before we can reify infinite uniform total strategies to infinitary analytic proofs:

Proposition 5.3.4 *In WSN, each uniform total strategy on* $[\![X; \Theta \vdash \Gamma]\!]$ *is the denotation of a unique infinitary analytic proof.*

The details follow those given in Sections 3.7 and 4.4: for proving some propositions it is convenient to view the P_ω core rule as an infinitely wide rule with countably many premises of $X; \Theta \vdash \bot, \Gamma$. Again, while analytic proofs are infinite they are 'productive': each finite portion of the (total) strategy can be found by examining a finite part of the proof tree.

We can thus normalise proofs in WSN to infinitary analytic proofs; and two proofs are denotationally equivalent if and only if they have the same normal form.

5.4 A Total Call-by-name Language

Next, we will show how we can embed a total call-by-name imperative language with an infinite ground type in WSN. This language might be compared to Gödel's System **T** with some imperative features and call-by-name semantics. The embedding will be based on negative formulas and the Intuitionistic Linear Logic translation.

5.4.1 Programming Language

Types and Terms

The language *TotLang* will be an applied simply-typed lambda calculus with base types `nat` (natural numbers), `com` (commands) and `var` (natural number reference cells).

Remark This language does not have products. Instead, we can represent a stateful object with methods of type `A` and `B` as a term `newobj : ((A -> B -> com) -> com` the idea is that `newobj (λ x y . c)` provides `c` with access to an instance of this object and its methods `x : A` and `y : B` following Idealized Algol [71]. We write `A * ... * AN` for the stateful object type `((A1 -> ... -> AN) -> com) -> com`.

The terms of TotLang are those of the simply typed lambda calculus over the given base types, together with the following constants (where `G` $\in$ `{ nat , com }`):

- **Imperative Flow:** `_ ; _ : com -> G -> G` , `skip : com`
- **Naturals:** `0 : nat` , `suc : nat -> nat`
- **Conditional and Looping:** `ifzero : nat -> G -> G -> G` , `repeat : nat -> com -> com`
- **Ground Reference Cells:** `_:=_ : var -> nat -> com` , `!_ : var -> nat` , `newvar : nat -> (var -> G) -> G` , `mkvar : nat -> (nat -> com) -> var`
- **Coroutines:** `_ || _ : (com -> com) -> (com -> com) -> com`
- **Encapsulation:** `encaps : (s -> s) * (s -> o) -> s -> (o -> com) -> com`

We will write `newvar x := n in M` as shorthand for `newvar n (λ x . M)`. The `mkvar` constructor allows the programmer to create custom variables that do not behave as a standard variable cell, they are required for the games model of TotLang to be fully abstract. The `encaps` operator acts as in the CBV setting, but as there are no products in TotLang we have needed to massage the types into a CPS form.

Example Programs

We briefly demonstrate the expressivity of TotLang.

- **Primitive Recursive Functions:** We can define addition as

  `λ m n . newvar x := n in repeat m (x := succ !x) ; !x`

  Similarly, we can define all of the primitive recursive functions.

- **Stack of Ground Values:** Using `encaps` we can define stacks of ground variables `newstack : (var -> com) -> com` using a similar approach to that in Section 5.2.1. Then `!` acts as `pop`; and `:=` as `push`.

```
λ f . encaps (λ h . newvar x := 0 in h a b) 0 f
      where a = λ n . ifzero !x then n else
                 (newvar z := !x - 1 in x := 0 ; !z)
            b = λ n . mkvar n (λ m . x := suc m)
```

- **Postfix Calculator**: We can use a stack to define a postfix calculator `newpfc : (nat -> (nat -> com) -> com -> com -> com) -> com` with answer, literal, addition and multiplication methods.

```
newpfc =  λ f . newstack (λ x . f ans lit add mult)
where     ans = !x
          lit = λ n . x := n
          add = x := !x + !x
          mult = x := !x × !x
```

- **Set of Naturals**: Similarly, we can define a `newset : ((nat -> com) -> (nat -> com) -> (nat -> nat) -> nat -> com) -> com` operator that constructs a set of naturals, where `newset (λ s.add s.rem s.elem s.count . H)` gives `H` access to the add, remove, test and count methods.

```
newset =  λ f . newstack (λ s . newvar n := 0 in f add rem elem count)
where     add = λ x . if (elem x) then () else (s := x ; n := n + 1)
          rem = λ x . newvar y := 0 in newstack t in
           repeat !n (let z = !s in
           if (z != x) then t := z else y := !y + 1) ;
           repeat (!n - !y) (s := !t) ; n := !n - !y
          elem = λ x . newvar y := 1 in newstack t in
           repeat !n (let z = !s in t := z; if (z == x) then y := 0) ;
           repeat !n (s := !t) ; !y
          count = !n
```

- **Growing Function**: We can use `encaps` for limited forms of higher-order state. For example, we can define `newgrow : (a -> a) -> ((a -> a) -> com) -> com` such that `newgrow f` creates a function that acts as `f` the first time it is called, `f` ∘ `f` the second time, and in general $\mathtt{f}^n$ on its nth interrogation.

Figure 5-5: TotLang types as WSN Formulas

$$\begin{array}{rcl} \mathsf{tlws}(\mathsf{com}) & = & \bot \lhd \top \\ \mathsf{tlws}(\mathsf{nat}) & = & \bot \lhd \overline{\omega} \\ \mathsf{tlws}(\mathsf{var}) & = & (\bot \lhd \overline{\omega}) \& (\omega \lhd \top) \\ \mathsf{tlws}(A \to B) & = & \mathsf{tlws}(B) \lhd ?\mathsf{tlws}(A)^{\bot} \end{array}$$

```
λ f g .  encaps (λ h .  h af bf) f g
    where af = λ j a .  j (f a)
          bf = λ j .  j
```

Thus we see that, while TotLang is a total programming language, it is nonetheless quite expressive.

We can give operational semantics to TotLang in a standard manner. The rules for `encaps` are: `E[encaps g a (λ x .  V)]` $\to$ `E[V]` and `E[encaps g a (λ x .  F(x)[x])]` $\to$ `E[g (λ j k .  encaps g (j a) (λ x .  F(x)[k a]))]`.

5.4.2 Embedding into WSN

Types

We translate types of TotLang to negative formulas of WSN using the function tlws given in Figure 5-5. A term $x_1 : A_1, \ldots, x_n : A_n \vdash s : B$ will be interpreted as a proof of $\vdash \mathsf{tlws}(B), ?\mathsf{tlws}(A_1)^{\bot}, \ldots, ?(A_n)^{\bot}$.

Terms : Lambda Calculus

For the λ-calculus fragment, we use our interpretation of Intuitionistic Linear Logic together with a Kleisli translation. Variables and abstraction are translated as follows:

$$\mathsf{P}^{+}_{\mathsf{wk}} \dfrac{\mathsf{P}^{?}_{\mathsf{der}} \dfrac{\mathsf{P}_{\mathsf{id}} \dfrac{}{\vdash A_i, A_i^{\bot}}}{\vdash A_i, ?A_i^{\bot}}}{\vdash A_i, ?A_1^{\bot}, \ldots, ?A_n^{\bot}} \qquad \dfrac{\vdash C, ?B^{\bot}, ?A_1^{\bot}, \ldots, ?A_n^{\bot}}{\vdash C \lhd ?B^{\bot}, ?A_1^{\bot}, \ldots, ?A_n^{\bot}}$$

The application rule is translated using cut and promotion as follows:

$$\mathsf{P}_{\mathsf{cut}}\,\dfrac{\vdash C \lhd ?B^{\perp}, ?A_1^{\perp}, \ldots, ?A_n^{\perp} \qquad \mathsf{P}^{+}_{\mathsf{sym}}\,\dfrac{\mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash C, C^{\perp}} \qquad \mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash !B, ?B^{\perp}}}{\vdash C, C^{\perp} \oslash !B, ?B^{\perp}}}{\vdash C, ?B^{\perp}, C^{\perp} \oslash !B}}{\mathsf{P}_{\mathsf{sym}}\,\dfrac{\vdash C, ?B, ?A_1^{\perp}, \ldots, ?A_n^{\perp}}{\vdash C, ?A_1^{\perp}, \ldots, ?A_n^{\perp}, ?B}} \qquad \mathsf{P}_{\mathsf{prom}}\,\dfrac{\vdash B, ?A_1^{\perp}, \ldots, ?A_n^{\perp}}{\vdash !B, ?A_1^{\perp}, \ldots, ?A_n^{\perp}}}{\mathsf{P}^{?}_{\mathsf{con}}\,\dfrac{\vdash C, ?A_1^{\perp}, \ldots, ?A_n^{\perp}, ?A_1^{\perp}, \ldots, ?A_n^{\perp}}{\vdash C, ?A_1^{\perp}, \ldots, ?A_n^{\perp}}}$$

Terms : Constants

We next give an interpretation of each of the constants. We write $\mathsf{P}^{+}_{\mathsf{wk}i}$ for the $\mathsf{P}^{+}_{\mathsf{wk}}$ rule that concludes $\vdash A_1, \ldots, A_n$ from $\vdash A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n$ for $1 \leqslant i \leqslant n$ where A_i is positive.

- `skip : com`, `zero : nat` and `suc : nat -> nat` are given by

$$\dfrac{\dfrac{}{\vdash \top}}{\vdash \perp, \top}\,\dfrac{}{\vdash \perp \lhd \top} \qquad \mathsf{P}_0\,\dfrac{\dfrac{}{\vdash \overline{\omega}}}{\vdash \perp, \overline{\omega}}\,\dfrac{}{\vdash \perp \lhd \overline{\omega}} \qquad \mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash \perp, \top} \qquad \mathsf{P}_{\mathsf{suc}}\,\dfrac{}{\vdash \omega, \overline{\omega}}}{\vdash \perp, \top \oslash \omega, \overline{\omega}}}{\dfrac{\vdash \perp, \overline{\omega}, \top \oslash \omega}{\vdash \perp \lhd \overline{\omega}, \top \oslash \omega}}$$

- `_ ; _ : com -> G -> G` is given as follows, where G ranges over $\perp, \omega$.

$$\mathsf{P}^{?}_{\mathsf{der}}\,\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash G, G^{\perp}}}{\vdash \top, G, G^{\perp}}}{\vdash \top \lhd G, G^{\perp}}}{\vdash \perp, G^{\perp}, (\top \lhd G)}}{\vdash \top, \perp, G^{\perp}, (\top \lhd G)}}{\vdash \top \lhd \perp, G^{\perp}, \top \lhd G}}{\vdash \perp, G^{\perp}, \top \lhd \perp, \top \lhd G}}{\vdash \perp \lhd G^{\perp}, \top \lhd \perp, \top \lhd G}}{\vdash \perp \lhd G^{\perp}, ?(\top \lhd \perp), ?(\top \lhd G)}$$

- `repeat : nat -> com -> com` is given as follows:

$$
\mathsf{P}^{?}_{\mathsf{der}}\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P}_{\mathsf{ind}}\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \top}}{\vdash \top, ?(\top \oslash \bot)}}{\vdash \top \parr ?(\top \oslash \bot)} \qquad \mathsf{P}^{\top}_{\parr}\dfrac{\mathsf{P}^{?}_{\mathsf{con}}\dfrac{\mathsf{P}_{\mathsf{mul}\otimes}\dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \top}}{\vdash \bot, \top}}{\vdash \top, \bot, \top}}{\vdash \top \oslash \bot, \top}}{\vdash \bot, \top \oslash \bot, \top}}{\vdash \bot, ?(\top \oslash \bot), \top}}{\vdash \bot, \top, ?(\top \oslash \bot)} \qquad \mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash !(\bot \lhd \top), ?(\top \oslash \bot)}}{\vdash \bot \otimes !(\bot \lhd \top), \top, ?(\top \oslash \bot), ?(\top \oslash \bot)}}{\vdash \bot \otimes !(\bot \lhd \top), \top, ?(\top \oslash \bot)}}{\vdash \bot \otimes !(\bot \lhd \top), \top \parr ?(\top \oslash \bot)}}{\vdash \omega, \top \parr ?(\top \oslash \bot)}}{\vdash \top, \omega, \top, ?(\top \oslash \bot)}}{\vdash \top \lhd \omega, \top, ?(\top \oslash \bot)}}{\vdash \bot, \top, \top \lhd \omega, ?(\top \oslash \bot)}}{\vdash \bot \oslash \top, \top \lhd \omega, ?(\top \oslash \bot)}}{\vdash \bot \oslash \top, ?(\top \lhd \omega), ?(\top \oslash \bot)}
$$

- `ifzero : nat -> G -> G -> G` is defined as follows, where G ranges over $\bot$, ω.

$$
\mathsf{P}^{?}_{\mathsf{der}}\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P}^{+}_{\mathsf{wk}3}\dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\dfrac{\mathsf{P}_{\multimap}\dfrac{\mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash \bot, \top} \qquad \mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash G, G^{\perp}}}{\vdash \bot, \top \oslash G, G^{\perp}}}{\vdash \bot, G^{\perp}, \top \oslash G}}{\vdash \bot, G^{\perp}, \top \oslash G, \top \oslash G} \qquad \mathsf{P}^{+}_{\mathsf{wk}2}\dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\dfrac{\mathsf{P}_{\multimap}\dfrac{\mathsf{P}_{\mathsf{ind}}\dfrac{\dfrac{}{\vdash \top} \qquad \dfrac{\dfrac{}{\vdash \top}}{\vdash \bot, \top}}{\vdash \omega, \top} \qquad \mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash G, G^{\perp}}}{\vdash \omega, \top \oslash G, G^{\perp}}}{\vdash \omega, G^{\perp}, \top \oslash G}}{\vdash \omega, G^{\perp}, \top \oslash G, \top \oslash G}}{\vdash \omega, G^{\perp}, \top \oslash G, \top \oslash G}}{\vdash \top, \omega, G^{\perp}, \top \oslash G, \top \oslash G}}{\vdash \bot, G^{\perp}, \top \oslash \omega, \top \oslash G, \top \oslash G}}{\vdash \bot \lhd G^{\perp}, \top \oslash \omega, \top \oslash G, \top \oslash G}}{\vdash \bot \lhd G^{\perp}, ?(\top \oslash \omega), ?(\top \oslash G), ?(\top \oslash G)}
$$

- `! _ : var -> nat` and `_ := _ : var -> nat -> com` are defined thus:

$$
\mathsf{P}^{?}_{\mathsf{der}}\dfrac{\mathsf{P}^{\top}_{\oplus 1}\dfrac{\mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash \bot \lhd \overline{\varpi}, \top \oslash \omega}}{\vdash \bot \lhd \overline{\varpi}, \top \oslash \omega \oplus \overline{\varpi} \oslash \bot}}{\vdash \bot \lhd \overline{\varpi}, ?(\top \oslash \omega \oplus \overline{\varpi} \oslash \bot)}
$$

$$
\mathsf{P}^{?}_{\mathsf{der}}\dfrac{\mathsf{P}^{\top}_{\oplus 2}\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P}_{\multimap}\dfrac{\mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash \omega, \overline{\varpi}} \qquad \mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash \bot, \top}}{\vdash \omega, \overline{\varpi} \oslash \bot, \top}}{\vdash \top, \omega, \overline{\varpi} \oslash \bot, \top}}{\vdash \top \oslash \omega, \overline{\varpi} \oslash \bot, \top}}{\vdash \bot, \top, \overline{\varpi} \oslash \bot, \top \oslash \omega}}{\vdash \bot \lhd \top, \overline{\varpi} \oslash \bot, \top \oslash \omega}}{\vdash \bot \lhd \top, \top \oslash \omega \oplus \overline{\varpi} \oslash \bot, \top \oslash \omega}}{\vdash \bot \lhd \top, ?(\top \oslash \omega \oplus \overline{\varpi} \oslash \bot), ?(\top \oslash \omega)}
$$

- `newvar : nat -> (var -> G) -> G` is translated as follows:

$$\mathsf{P}^{?}_{\mathsf{der}}\,\dfrac{\mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{ana}}\,\dfrac{\dfrac{p_r \vdash \bot \lhd \overline{\omega}, \bot \lhd \overline{\omega}, \top \oslash \omega \qquad p_w \vdash \omega \lhd \top, \bot \lhd \overline{\omega}, \top \oslash \omega}{\vdash \bot \lhd \overline{\omega} \& \omega \lhd \top, \bot \lhd \overline{\omega}, \top \oslash \omega}}{\vdash !\mathbf{var}, \top \oslash \omega}}{\vdash G, G^{\perp} \oslash !\mathbf{var}, \top \oslash \omega}}{\vdash G, ?(G^{\perp} \oslash !\mathbf{var}), ?(\top \oslash \omega)}$$

where G ranges over the formula representation of ground types, p_r is:

$$\dfrac{\dfrac{\dfrac{\mathsf{P}_{\mathsf{ind}}\,\dfrac{\dfrac{\mathsf{P}^{0}_{\overline{\omega}}\,\dfrac{\dfrac{\dfrac{\mathsf{P}^{0}_{\overline{\omega}}\,\dfrac{\dfrac{}{\vdash \top}}{\vdash \overline{\omega}}}{\vdash \bot \lhd \overline{\omega}}}{\vdash \top, \bot \lhd \overline{\omega}}}{\vdash \overline{\omega}, \bot \lhd \overline{\omega}}}{\vdash \overline{\omega} \oslash (\bot \lhd \overline{\omega})} \qquad \dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\,\dfrac{\mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{suc}}\,\dfrac{}{\vdash \omega, \overline{\omega}} \qquad \dfrac{\dfrac{\mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash \bot, \top} \qquad \mathsf{P}_{\mathsf{suc}}\,\dfrac{}{\vdash \omega, \overline{\omega}}}{\vdash \bot, \top \oslash \omega, \overline{\omega}}}{\vdash \bot, \overline{\omega}, \top \oslash \omega}}{\vdash \bot \lhd \overline{\omega}, \top \oslash \omega}}{\vdash \omega, \overline{\omega} \oslash (\bot \lhd \overline{\omega}), (\top \oslash \omega)}}{\vdash \omega, (\top \oslash \omega), \overline{\omega} \oslash (\bot \lhd \overline{\omega})}}{\vdash \omega \lhd (\top \oslash \omega), \overline{\omega} \oslash (\bot \lhd \overline{\omega})}}{\vdash \omega, \overline{\omega} \oslash (\bot \lhd \overline{\omega})}}{\vdash \top \oslash \omega, \overline{\omega} \oslash (\bot \lhd \overline{\omega})}}{\vdash \bot, \overline{\omega}, \bot \lhd \overline{\omega}, \top \oslash \omega}}{\vdash \bot \lhd \overline{\omega}, \bot \lhd \overline{\omega}, \top \oslash \omega}$$

and p_w is:

$$\mathsf{P}^{+}_{\mathsf{wk}}\,\dfrac{\dfrac{\mathsf{P}^{\mathsf{lfe}}_{\omega}\,\dfrac{\mathsf{P}_{\mathsf{ind}}\,\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{P}_{0}\,\dfrac{}{\vdash \overline{\omega}}}{\vdash \bot \lhd \overline{\omega}}}{\vdash \top, \bot \lhd \overline{\omega}}}{\vdash \top \oslash (\bot \lhd \overline{\omega})} \qquad \dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\,\dfrac{\mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash \bot, \top} \qquad \dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\,\dfrac{\mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash \bot, \top} \qquad \mathsf{P}_{\mathsf{suc}}\,\dfrac{}{\vdash \omega, \overline{\omega}}}{\vdash \bot, \top \oslash \omega, \overline{\omega}}}{\vdash \bot, \overline{\omega}, \top \oslash \omega}}{\vdash \bot \lhd \overline{\omega}, \top \oslash \omega}}{\vdash \bot, \top \oslash (\bot \lhd \overline{\omega}), \top \oslash \omega}}{\vdash \bot, \top \oslash \omega, \top \oslash (\bot \lhd \overline{\omega})}}{\vdash \bot \lhd (\top \oslash \omega), \top \oslash (\bot \lhd \overline{\omega})}}{\vdash \omega, \top \oslash (\bot \lhd \overline{\omega})}}{\vdash \omega, \top, \bot \lhd \overline{\omega}}}{\vdash \omega \lhd \top, \bot \lhd \overline{\omega}}}{\vdash \omega \lhd \top, \bot \lhd \overline{\omega}, \top \oslash \omega}$$

- `mkvar :  (nat -> com) -> nat -> var`

$$\mathsf{P}^{?}_{\mathsf{der}}\,\dfrac{\mathsf{P}^{+}_{\mathsf{wk}}\,\dfrac{\mathsf{P}^{\mathsf{T}}_{\parr}\,\dfrac{\mathsf{P}_{\mathsf{ind}}\,\dfrac{p' \qquad p''}{\vdash \omega, \top \parr ((\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega}))}}{\vdash \omega, \top, (\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega})}}{\vdash \omega \lhd \top, (\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega})}}{\vdash \omega \lhd \top, \top \oslash \omega, (\top \oslash \bot) \lhd !(\bot \oslash \overline{\omega})} \qquad \mathsf{P}^{+}_{\mathsf{wk}}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash \bot \lhd \overline{\omega}, \top \oslash \omega}}{\vdash \bot \lhd \overline{\omega}, \top \oslash \omega, (\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega})}}{\vdash \bot \lhd \overline{\omega} \& \omega \lhd \top, \top \oslash \omega, (\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega})}$$

where p' is:

$$\mathsf{P}_{\mathsf{mul}\otimes}\,\dfrac{\dfrac{}{\vdash \top}}{\vdash \bot, \top} \qquad \mathsf{P}_{\mathsf{prom}}\,\dfrac{\mathsf{P}_0\,\dfrac{}{\vdash \overline{\omega}}}{\vdash \bot \lhd \overline{\omega}}}{\vdash !(\bot \lhd \overline{\omega})}}{\vdash \bot \otimes !(\bot \lhd \overline{\omega}), \top}}{\vdash \top, \bot, !(\bot \lhd \overline{\omega}), \top}}{\vdash (\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega}), \top}}{\vdash \top \parr ((\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega}))}$$

and p'' is:

$$\mathsf{P}^{\mathsf{T}}_{\parr}\,\dfrac{\mathsf{P}_{\mathsf{mul}\otimes}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash \bot, \top} \qquad \mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash \bot \lhd \top, \top \oslash \bot} \qquad \mathsf{P}_{\mathsf{prom}}\,\dfrac{\mathsf{P}^{?}_{\mathsf{der}}\,\dfrac{\mathsf{P}^{+}_{\mathsf{sym}}\,\dfrac{\mathsf{P}_{\multimap}\,\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\vdash \bot, \top} \qquad \mathsf{P}_{\mathsf{suc}}\,\dfrac{}{\vdash \omega, \overline{\omega}}}{\vdash \bot, \top \oslash \omega, \overline{\omega}}}{\vdash \bot, \overline{\omega}, \top \oslash \omega}}{\vdash \bot \lhd \overline{\omega}, ?(\top \oslash \omega)}}{\vdash !(\bot \lhd \overline{\omega}), ?(\top \oslash \omega)}}{\vdash \bot \lhd \top, (\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega}), ?(\top \oslash \omega)}}{\vdash \bot \lhd \top, ?(\top \oslash \omega), (\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega})}}{\vdash \bot \otimes ((\bot \lhd \top) \lhd ?(\top \oslash \omega)), \top, ((\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega}))}}{\vdash \bot \otimes ((\bot \lhd \top) \lhd ?(\top \oslash \omega)), \top \parr ((\top \oslash \bot) \oslash !(\bot \lhd \overline{\omega}))}$$

- We translate `_ || _ : (com -> com) -> (com -> com) -> com` using cocomp as defined in Section 5.2.2.

$$\mathsf{P}^{?}_{\mathsf{der}}\,\dfrac{\mathsf{cocomp} \vdash \Sigma, \Sigma^{\perp} \oslash !\Sigma, \Sigma^{\perp} \oslash !\Sigma}{\vdash \Sigma, ?(\Sigma^{\perp} \oslash !\Sigma), ?(\Sigma^{\perp} \oslash !\Sigma)}}{\vdash (\Sigma \lhd ?(\Sigma^{\perp} \oslash !\Sigma)) \lhd ?(\Sigma^{\perp} \oslash !\Sigma)}$$

- We next describe `encaps : (((a -> a) -> (a -> b) -> com) -> com) -> a -> (b -> com) -> com`. We use the anamorphism rule to construct a function $a \to !b$ and compose this with the other inputs. The CPS types here makes the embedding somewhat cumbersome. We introduce the unary proof rule $\mathsf{P}_{\multimap \mathsf{id}} = \mathsf{P}_{\multimap}(\mathsf{P}_{\mathsf{id}}, \_)$.

$$
\begin{array}{ll}
\mathsf{P}_{\mathsf{id}} & \overline{\vdash !A, ?A^\perp} \\
\mathsf{P}_{\multimap\mathsf{id}} & \overline{\vdash !B, ?B^\perp \oslash !A, ?A^\perp} \\
\mathsf{P}_{\multimap\mathsf{id}} & \overline{\vdash \Sigma, \Sigma^\perp \oslash !B, ?B^\perp \oslash !A, ?A^\perp} \\
\mathsf{P}^{+}_{\mathsf{sym}} & \overline{\vdash \Sigma, ?B^\perp \oslash !A, \Sigma^\perp \oslash !B, ?A^\perp} \\
 & \overline{\vdash \Sigma \lhd (?B^\perp \oslash !A), ?(\Sigma^\perp \oslash !B), ?A^\perp} \\
\mathsf{P}_{\mathsf{prom}} & \overline{\vdash !(\Sigma \lhd (?B^\perp \oslash !A)), ?(\Sigma^\perp \oslash !B), ?A^\perp} \\
\mathsf{P}_{\multimap\mathsf{id}} & \overline{\vdash \Sigma, \Sigma^\perp \oslash !(\Sigma \lhd (?B^\perp \oslash !A)), ?(\Sigma^\perp \oslash !B), ?A^\perp} \\
\mathsf{P}^{+}_{\mathsf{sym}} & \overline{\vdash \Sigma, ?(\Sigma^\perp \oslash !B), ?A^\perp, \Sigma^\perp \oslash !(\Sigma \lhd (?B^\perp \oslash !A))} \qquad p_1 \\
\mathsf{P}_{\mathsf{cut}} & \overline{\vdash \Sigma, ?(\Sigma^\perp \oslash !B), ?A^\perp, \Sigma^\perp \oslash !(\Sigma \lhd ?(A^\perp \parr B^\perp \oslash !A))} \qquad p_2 \\
\mathsf{P}_{\mathsf{cut}} & \overline{\vdash \Sigma, ?(\Sigma^\perp \oslash !B), ?A^\perp, \Sigma^\perp \oslash !((\Sigma \lhd ?(A^\perp \oslash !A)) \lhd ?(B^\perp \oslash !A))} \\
\mathsf{P}^{?}_{\mathsf{der}} & \overline{\vdash \Sigma, ?(\Sigma^\perp \oslash !B), ?A^\perp, ?(\Sigma^\perp \oslash !((\Sigma \lhd ?(A^\perp \oslash !A)) \lhd ?(B^\perp \oslash !A)))}
\end{array}
$$

where p_1 is:

$$
\begin{array}{ll}
\mathsf{P}_{\mathsf{id}} & \overline{\vdash A, A^\perp} \\
\mathsf{P}^{+}_{\mathsf{wk}} & \overline{\vdash A, A^\perp, B^\perp} \\
\mathsf{P}^{\mathsf{T}}_{\parr} & \overline{\vdash A, A^\perp \parr B^\perp} \qquad \mathsf{P}_{\mathsf{id}}\ \overline{\vdash !A, ?A^\perp} \\
\mathsf{P}_{\multimap} & \overline{\vdash A, A^\perp \parr B^\perp \oslash !A, ?A^\perp} \\
\mathsf{P}^{+}_{\mathsf{sym}} & \overline{\vdash A, ?A^\perp, A^\perp \parr B^\perp \oslash !A} \\
\mathsf{P}^{?}_{\mathsf{der}} & \overline{\vdash A, ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}_{\mathsf{prom}} & \overline{\vdash !A, ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A)} \qquad \mathsf{P}_{\mathsf{id}}\ \overline{\vdash !(A \otimes B \lhd ?A^\perp), ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}_{\mathsf{mul}\otimes} & \overline{\vdash !A \otimes !(A \otimes B \lhd ?A^\perp), ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A), ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}^{?}_{\mathsf{con}} & p' \qquad \overline{\vdash !A \otimes !(A \otimes B \lhd ?A^\perp), ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}_{\mathsf{mul}} & \overline{\vdash B, !A \otimes !(A \otimes B \lhd ?A^\perp), ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A), ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}^{+}_{\mathsf{sym}} & \overline{\vdash B, !A \otimes !(A \otimes B \lhd ?A^\perp), ?A^\perp, ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A), ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}^{?}_{\mathsf{con}} & \overline{\vdash B, !A \otimes !(A \otimes B \lhd ?A^\perp), ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}^{\mathsf{T}}_{\parr} & \overline{\vdash B, !A \otimes !(A \otimes B \lhd ?A^\perp), ?A^\perp \parr ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}_{\mathsf{ana}} & \overline{\vdash !B, ?A^\perp \parr ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}^{\mathsf{T}}_{\parr} & \overline{\vdash !B \lhd ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}_{\multimap\mathsf{id}} & \overline{\vdash \Sigma, \Sigma^\perp \oslash (!B \lhd ?A^\perp), ?(A^\perp \parr B^\perp \oslash !A)} \\
\mathsf{P}^{+}_{\mathsf{sym}} & \overline{\vdash \Sigma, ?(A^\perp \parr B^\perp \oslash !A), \Sigma^\perp \oslash (!B \lhd ?A^\perp)} \\
 & \overline{\vdash \Sigma \lhd ?(A^\perp \parr B^\perp \oslash !A), \Sigma^\perp \oslash (!B \lhd ?A^\perp)} \\
\mathsf{P}^{?}_{\mathsf{der}} & \overline{\vdash \Sigma \lhd ?(A^\perp \parr B^\perp \oslash !A), ?(\Sigma^\perp \oslash (!B \lhd ?A^\perp))} \\
\mathsf{P}_{\mathsf{prom}} & \overline{\vdash !(\Sigma \lhd ?(A^\perp \parr B^\perp \oslash !A)), ?(\Sigma^\perp \oslash (!B \lhd ?A^\perp))} \\
\mathsf{P}_{\multimap\mathsf{id}} & \overline{\vdash \Sigma, \Sigma^\perp \oslash !(\Sigma \lhd ?(A^\perp \parr B^\perp \oslash !A)), ?(\Sigma^\perp \oslash (!B \lhd ?A^\perp))} \\
\mathsf{P}^{+}_{\mathsf{sym}} & \overline{\vdash \Sigma, ?(\Sigma^\perp \oslash (!B \lhd ?A^\perp)), \Sigma^\perp \oslash !(\Sigma \lhd ?(A^\perp \parr B^\perp \oslash !A))} \\
 & \overline{\vdash \Sigma \lhd ?(\Sigma^\perp \oslash (!B \lhd ?A^\perp)), \Sigma^\perp \oslash !(\Sigma \lhd ?(A^\perp \parr B^\perp \oslash !A))}
\end{array}
$$

where p' is:

$$
\begin{prooftree}
\AxiomC{}
\LeftLabel{$\mathsf{P_{id}}$}
\UnaryInfC{$\vdash B, B^\perp$}
\LeftLabel{$\mathsf{P^{+}_{wk}}$}
\UnaryInfC{$\vdash B, A^\perp, B^\perp$}
\LeftLabel{$\mathsf{P^{\mathsf{T}}_{\parr}}$}
\UnaryInfC{$\vdash B, A^\perp \parr B^\perp$}
\AxiomC{}
\LeftLabel{$\mathsf{P_{id}}$}
\UnaryInfC{$\vdash !A, ?A^\perp$}
\LeftLabel{$\mathsf{P_{\multimap}}$}
\BinaryInfC{$\vdash B, A^\perp \parr B^\perp \oslash !A, ?A^\perp$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash B, ?A^\perp, A^\perp \parr B^\perp \oslash !A$}
\LeftLabel{$\mathsf{P^{?}_{der}}$}
\UnaryInfC{$\vdash B, ?A^\perp, ?(A^\perp \parr B^\perp \oslash !A)$}
\end{prooftree}
$$

and p_2 is:

$$
\begin{prooftree}
\AxiomC{}
\LeftLabel{$\mathsf{P_{id}}$}
\UnaryInfC{$\vdash !A, ?A^\perp$}
\LeftLabel{$\mathsf{P_{\multimap id}}$}
\UnaryInfC{$\vdash A, A^\perp \oslash !A, ?A^\perp$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash A, ?A^\perp, A^\perp \oslash !A$}
\LeftLabel{$\mathsf{P^{?}_{der}}$}
\UnaryInfC{$\vdash A, ?A^\perp, ?(A^\perp \oslash !A)$}
\AxiomC{}
\LeftLabel{$\mathsf{P_{id}}$}
\UnaryInfC{$\vdash !A, ?A^\perp$}
\LeftLabel{$\mathsf{P_{\multimap id}}$}
\UnaryInfC{$\vdash B, B^\perp \oslash !A, ?A^\perp$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash B, ?A^\perp, B^\perp \oslash !A$}
\LeftLabel{$\mathsf{P^{?}_{der}}$}
\UnaryInfC{$\vdash B, ?A^\perp, ?(B^\perp \oslash !A)$}
\LeftLabel{$\mathsf{P_{mul\otimes}}$}
\BinaryInfC{$\vdash A \otimes B, ?A^\perp, ?(A^\perp \oslash !A), ?A^\perp, ?(B^\perp \oslash !A)$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash A \otimes B, ?A^\perp, ?A^\perp, ?(A^\perp \oslash !A), ?(B^\perp \oslash !A)$}
\LeftLabel{$\mathsf{P^{?}_{con}}$}
\UnaryInfC{$\vdash A \otimes B, ?A^\perp, ?(A^\perp \oslash !A), ?(B^\perp \oslash !A)$}
\UnaryInfC{$\vdash A \otimes B \lhd ?A^\perp, ?(A^\perp \oslash !A), ?(B^\perp \oslash !A)$}
\LeftLabel{$\mathsf{P_{prom}}$}
\UnaryInfC{$\vdash !(A \otimes B \lhd ?A^\perp), ?(A^\perp \oslash !A), ?(B^\perp \oslash !A)$}
\LeftLabel{$\mathsf{P_{\multimap id}}$}
\UnaryInfC{$\vdash \Sigma, \Sigma^\perp \oslash !(A \otimes B \lhd ?A^\perp), ?(A^\perp \oslash !A), ?(B^\perp \oslash !A)$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash \Sigma, ?(A^\perp \oslash !A), ?(B^\perp \oslash !A), \Sigma^\perp \oslash !(A \otimes B \lhd ?A^\perp)$}
\UnaryInfC{$\vdash (\Sigma \lhd ?(A^\perp \oslash !A)) \lhd ?(B^\perp \oslash !A), \Sigma^\perp \oslash !(A \otimes B \lhd ?A^\perp)$}
\LeftLabel{$\mathsf{P^{?}_{der}}$}
\UnaryInfC{$\vdash (\Sigma \lhd ?(A^\perp \oslash !A)) \lhd ?(B^\perp \oslash !A), ?(\Sigma^\perp \oslash !(A \otimes B \lhd ?A^\perp))$}
\LeftLabel{$\mathsf{P_{prom}}$}
\UnaryInfC{$\vdash !((\Sigma \lhd ?(A^\perp \oslash !A)) \lhd ?(B^\perp \oslash !A)), ?(\Sigma^\perp \oslash !(A \otimes B \lhd ?A^\perp))$}
\LeftLabel{$\mathsf{P_{\multimap id}}$}
\UnaryInfC{$\vdash \Sigma, \Sigma^\perp \oslash !((\Sigma \lhd ?(A^\perp \oslash !A)) \lhd ?(B^\perp \oslash !A)), ?(\Sigma^\perp \oslash !(A \otimes B \lhd ?A^\perp))$}
\LeftLabel{$\mathsf{P^{+}_{sym}}$}
\UnaryInfC{$\vdash \Sigma, ?(\Sigma^\perp \oslash !(A \otimes B \lhd ?A^\perp)), \Sigma^\perp \oslash !((\Sigma \lhd ?(A^\perp \oslash !A)) \lhd ?(B^\perp \oslash !A))$}
\UnaryInfC{$\vdash \Sigma \lhd ?(\Sigma^\perp \oslash !(A \otimes B \lhd ?A^\perp)), \Sigma^\perp \oslash !((\Sigma \lhd ?(A^\perp \oslash !A)) \lhd ?(B^\perp \oslash !A))$}
\end{prooftree}
$$

Literature Comparison

This model is based on the ILL + Kleisli embedding. Ground state works as in [9]: in the interpretation of `newvar`, the application of the map `var -> G` to the Boolean cell of type $!\mathsf{var}$ does not implicitly promote its argument: multiple occurrences of the same variable *share* the underlying strategy via contraction. This pattern is also found in the interpretation of `encaps`.

Full Abstraction

We conjecture that the model of TotLang inside $\mathcal{W}$, via our WSN embedding, is fully abstract — i.e. two terms are observationally equivalent if and only if they are mapped to the same strategy, following [8] and [51]. From this it follows that M and N are observationally equivalent if and only if $[\![\mathsf{tlws}(M)]\!] = [\![\mathsf{tlws}(N)]\!]$ which holds if and only if $\mathsf{tlws}(M)$ and $\mathsf{tlws}(N)$ have the same (infinitary) normal forms.

5.5 Properties of Programs

The formulas available in $\mathsf{WS1}$ are more expressive than the types in the languages described above. We can exploit this by giving types (formulas) to terms (proofs) that specify more behavioural properties than their basic type. We gave a "good variable" example in Section 4.1.4, here we take a more general approach. We focus on the call-by-name embedding.

Note that the expressivity here is restricted to *specifying* properties. Satisfaction of these properties can be determined by examining the semantics. We may also wish to consider syntactic ways of showing that a program satisfies these properties, but this lies outside the scope of this thesis.

5.5.1 Using the First-order Structure

We wish to use the first-order structure of $\mathsf{WS1}$ to apply predicates to the ground values that are provided as the inputs and outputs of programs. To do this, we can exploit the specialisation of $\mathsf{WS1}$ given in Section 4.6. In this setting the underlying model is fixed (natural numbers), and we assume constants 0 and s denoting the zero value and successor function. There is an additional induction rule given by:

$$\mathsf{P}'_{\mathsf{ind}}\ \frac{X;\Theta \vdash N[0/x] \qquad X \uplus \{x\};\Theta \vdash N[s(x)/x], N^{\perp}}{X;\Theta \vdash \forall x.N}$$

Further, we can add additional function symbols and predicates to the first-order language as desired, such as $\leqslant$ and a positive version of $=$.

In this setting, the translation of the `nat` data type $\bot \lhd \overline{\omega}$ denotes the same game as $\bot \lhd \exists x.\top$. We can show that $\forall x.\bot \cong \omega$:

$$\dfrac{\dfrac{\mathsf{P}_{\mathsf{id}}\,\dfrac{}{\{x\}\vdash\omega,\overline{\omega}}}{\mathsf{P}^{x}_{\exists}\,\dfrac{\dfrac{\dfrac{\{x\}\vdash\top,\omega,\overline{\omega}}{\{x\}\vdash\top\oslash\omega,\overline{\omega}}}{\{x\}\vdash\exists x.(\top\oslash\omega),\overline{\omega}}}{\{x\}\vdash\bot,\overline{\omega},\exists x.(\top\oslash\omega)}}{\vdash\forall x.\bot,\overline{\omega},\exists x.(\top\oslash\omega)}\qquad \mathsf{P}'_{\mathsf{ind}}\,\dfrac{\mathsf{P}_0\,\dfrac{}{\vdash\overline{\omega}}}{\vdash\bot,\overline{\omega}}\qquad\dfrac{\mathsf{P}_{\mathsf{suc}}\,\dfrac{}{\{x\}\vdash\omega,\overline{\omega}}}{\{x\}\vdash\top,\omega,\overline{\omega}}}{\vdash\bot\lhd\overline{\omega}}\qquad\dfrac{\dfrac{\{x\}\vdash\top\oslash\omega,\overline{\omega}}{\{x\}\vdash\bot,\overline{\omega},\top\oslash\omega}}{\{x\}\vdash\bot\lhd\overline{\omega},\top\oslash\omega}}{\vdash\forall x.(\bot\lhd\overline{\omega})}}{p_{\forall\overline{\omega}}\vdash\forall x.\bot,\overline{\omega}}\,\mathsf{P}_{\mathsf{cut}}$$

$$\dfrac{\mathsf{P}^{0}_{\exists}\,\dfrac{}{\vdash\top}}{\vdash\exists x.\top}\qquad\dfrac{\mathsf{P}^{s(x)}_{\exists}\,\dfrac{\dfrac{}{\{x\}\vdash\top}}{\{x\}\vdash\exists x.\top}}{\{x\}\vdash\bot,\exists x.\top}}{\vdash\forall x.\bot,\exists x.\top}}{p_{\omega\exists}\vdash\omega,\exists x.\top}\,\mathsf{P}_{\mathsf{ind}}$$

The denotation of the above proofs are the identity maps. By utilising this isomorphism we will be able to use the first-order structure to represent properties on TotLang programs.

5.5.2 Embeddings and Specifications

Given a formula M (which may be the translation of TotLang type) and a proof p of $\vdash M$ (which may be the translation of a TotLang program) we next describe what it means for p to satisfy a specification S on M. A specification S will be a syntactic object which represents a subgame of M: a proof p satisfies S if all of the plays in $[\![p]\!]$ lie within $[\![S]\!]$. We have already encountered the notion of subgame when considering uniformity of strategies — a subgame of M consists of a game N and an embedding $N \rightharpoonup M$: a pair of strategies $\mathsf{in} : N \multimap M$ and $\mathsf{out} : M \multimap N$ such that $\mathsf{out} \circ \mathsf{in} = \mathsf{id}$ and $\mathsf{in} \circ \mathsf{out} \sqsubseteq \mathsf{id}$.

Definition Let M be a negative formula of WS1. A *specification* on M is a negative formula S together with an embedding $[\![S]\!] \rightharpoonup [\![M]\!]$.

We say that $p \vdash M$ *satisfies* the specification $(S, \mathsf{in}, \mathsf{out})$ if each play in $[\![p]\!]$ lies within S — i.e. $[\![p]\!] = \mathsf{in} \circ \tau$ for some $\tau : S$, or that $[\![p]\!]$ *factors through* the embedding.

Note that an embedding may not be total: as we have seen, $\mathbf{1}$ is a subgame of $\bot$, but the map $\mathsf{in} : \mathbf{1} \multimap \bot$ is not total. If it is total, we can ask if it is definable by some WS1 proof.

Definition A specification $(\mathsf{in}, \mathsf{out}) : [\![S]\!] \rightharpoonup [\![M]\!]$ is *definable* if there is a proof $p \vdash M, S^{\perp}$ with $[\![p]\!] = \mathsf{in}$.

If a specification is definable, we can construct a proof of $\vdash M$ for any proof of the corresponding specification $\vdash S$ using $\mathsf{P}_{\mathsf{cut}}$.

5.5.3 Program Specifications

In the next section we will give some concrete examples of specifications on program types with respect to TotLang; we will first identify some general patterns. We first consider some definable embeddings.

- $M, P \lhd Q, \Delta \rightharpoonup M, P, Q, \Delta$ — This corresponds to a specification regarding order of arguments, requiring that the proof/program accesses P before it accesses Q. To give the embedding, we use $\mathsf{P}_{\multimap}$ together with a proof of

$$\vdash P^{\perp} \oslash Q^{\perp}, P \parr Q$$

  using $\mathsf{P}_{\mathsf{mul}}$ and P_{id}.

- $M, (P \lhd \_)^n P, \Delta \rightharpoonup M, ?P, \Delta$ — This corresponds to a specification regarding the number of times an argument can be interrogated. In particular, if a proof of $M, ?P, \Delta$ factors through this specification it must access its argument in P at most $n+1$ times. To give the embedding, we use $\mathsf{P}_{\multimap}$ together with a proof of

$$\vdash (P^{\perp} \oslash \_)^n (P^{\perp}), ?P$$

  which makes n uses of contraction, dereliction, and $\mathsf{P}_{\mathsf{mul}}$.

- $\perp \lhd \exists x.\top \rightharpoonup \perp \lhd \overline{\omega}$ — Following the isomorphism given in Section 5.5.1, we can replace an instance of the natural number type by a formula that binds the value played to a first-order variable.

- $M, \overline{\phi}(\overrightarrow{s}), \Delta \rightharpoonup M, \top, \Delta$ — when a move $\top$ is to be played, this specification ensures that the atomic formula $\overline{\phi}(\overrightarrow{s})$ is true in the model. The embedding uses $\mathsf{P}_{\multimap}$ together with the unique analytic proof of $\vdash \phi(\overrightarrow{s}), \top$.

- $M, P(n), \Delta \rightharpoonup M, \exists x.P(x), \Delta$ for each natural number n — this corresponds to a specification that requires the chosen value of x to be n. The embedding can be constructed using $\mathsf{P}_{\multimap}$ and a proof of

$$\vdash P(n)^{\perp}, \exists x.P(x)$$

  which uses $\mathsf{P}_{\exists}^{\mathsf{T}n}$ and P_{id}.

The following embedding is not definable by a proof (in is not total):

- $\alpha \lhd \top \rightharpoonup \perp \lhd \top$ — where α is a (nullary) atom. This uses an underlying map $\alpha \rightharpoonup \perp$ whose in component is given by $\mathsf{id}_{\perp}$ or ϵ, depending on the truth value of α. We can use this to control the order that moves are played, noting that moves

in negative occurrences of α must occur before moves in positive occurrences ($\overline{\alpha}$) following the example in Section 4.1.4.

Examples

We next give some concrete examples of $\mathsf{WS1}$ specifications on TotLang types. A specification on a TotLang type T is just a specification on $\mathsf{tlws}(T)$. In this section, we wish our quantifiers to range over the maximum scope, and so a formula $\forall x.\alpha \lhd \exists y.\overline{\beta} \oslash \forall z.\gamma \lhd \exists w.\overline{\delta}$ should be read as $\forall x.(\alpha \lhd \exists y.(\overline{\beta} \oslash \forall z.(\gamma \lhd \exists w.\overline{\delta})))$, where the only choice at each move is the appropriate value in the model.

- **Identity function:** We can give a specification on `nat -> nat` that is only satisfied by the identity function. This is given by $S = \bot \lhd \top \oslash \forall x.\bot \lhd \exists y.y = x$, read as $\bot \lhd (\top \oslash (\forall x.(\bot \lhd (\exists y.y = x))))$ — these moves correspond to output-request, input-request, input, output respectively. The embedding is given below:

$$\frac{}{\{y,z\}; y = z \vdash \top}$$

$$\mathsf{P}^{y}_{\exists}\ \frac{\{y,z\}; y = z \vdash \top}{\{y,z\}; y = z \vdash \exists u.\top}$$

$$\frac{\{y,z\}; y = z \vdash \exists u.\top}{\{y,z\}; y = z \vdash \bot, \exists u.\top}$$

$$\mathsf{P}_{\mathsf{cut}}\ \frac{\{y,z\}; y = z \vdash \bot, \exists u.\top \qquad p_{\forall\overline{\omega}} : \{y,z\}; y = z \vdash \forall u.\top, \overline{\omega}}{\{y,z\}; y = z \vdash \bot, \overline{\omega}}$$

$$\frac{\{y,z\}; y = z \vdash \bot, \overline{\omega}}{\{y,z\} \vdash y \neq z, \overline{\omega}}$$

$$\frac{\{y,z\} \vdash y \neq z, \overline{\omega}}{\{z\} \vdash \forall y.y \neq z, \overline{\omega}}$$

$$\frac{\{z\} \vdash \forall y.y \neq z, \overline{\omega}}{\{z\} \vdash \top, \forall y.y \neq z, \overline{\omega}}$$

$$\frac{\{z\} \vdash \top, \forall y.y \neq z, \overline{\omega}}{\{z\} \vdash \top \oslash \forall y.y \neq z, \overline{\omega}}$$

$$\mathsf{P}^{z}_{\exists}\ \frac{\{z\} \vdash \top \oslash \forall y.y \neq z, \overline{\omega}}{\{z\} \vdash \exists x.\top \oslash \forall y.y \neq x, \overline{\omega}}$$

$$\frac{\{z\} \vdash \exists x.\top \oslash \forall y.y \neq x, \overline{\omega}}{\{z\} \vdash \bot, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}$$

$$\frac{\{z\} \vdash \bot, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}{\vdash \forall z.\bot, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}$$

$$\mathsf{P}_{\mathsf{cut}}\ \frac{p_{\omega\exists} \vdash \omega, \exists z.\top \qquad \vdash \forall z.\bot, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}{\vdash \omega, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}$$

$$\frac{\vdash \omega, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}{\vdash \top, \omega, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}$$

$$\frac{\vdash \top, \omega, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}{\vdash \top \oslash \omega, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}$$

$$\frac{\vdash \top \oslash \omega, \overline{\omega}, \exists x.\top \oslash \forall y.y \neq x}{\vdash \overline{\omega} \parr (\top \oslash \omega), \exists x.\top \oslash \forall y.y \neq x}$$

$$\frac{\vdash \overline{\omega} \parr (\top \oslash \omega), \exists x.\top \oslash \forall y.y \neq x}{\vdash \bot, \exists x.\top \oslash \forall y.y \neq x, \overline{\omega} \parr (\top \oslash \omega)}$$

$$\frac{\vdash \bot, \exists x.\top \oslash \forall y.y \neq x, \overline{\omega} \parr (\top \oslash \omega)}{\vdash \bot \lhd (\exists x.\top \oslash \forall y.y \neq x), \overline{\omega} \parr (\top \oslash \omega)}$$

$$\frac{\vdash \bot \lhd (\exists x.\top \oslash \forall y.y \neq x), \overline{\omega} \parr (\top \oslash \omega)}{\vdash \top, \bot \lhd \exists x.\top \oslash \forall y.y \neq x, \overline{\omega} \parr (\top \oslash \omega)}$$

$$\frac{\vdash \top, \bot \lhd \exists x.\top \oslash \forall y.y \neq x, \overline{\omega} \parr (\top \oslash \omega)}{\vdash \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x, \overline{\omega} \parr (\top \oslash \omega)}$$

$$\frac{\vdash \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x, \overline{\omega} \parr (\top \oslash \omega)}{\vdash \bot, \overline{\omega}, \top \oslash \omega, \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x}$$

$$\frac{\vdash \bot, \overline{\omega}, \top \oslash \omega, \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x}{\vdash \bot \lhd \overline{\omega}, \top \oslash \omega, \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x}$$

$$\mathsf{P}^{?}_{\mathsf{der}}\ \frac{\vdash \bot \lhd \overline{\omega}, \top \oslash \omega, \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x}{\vdash \bot \lhd \overline{\omega}, ?(\top \oslash \omega), \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x}$$

$$\frac{\vdash \bot \lhd \overline{\omega}, ?(\top \oslash \omega), \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x}{\vdash (\bot \lhd \overline{\omega}) \lhd ?(\top \oslash \omega), \top \oslash \bot \lhd \exists x.\top \oslash \forall y.y \neq x}$$

The only strategy on $[\![S]\!]$ corresponds to the copycat strategy on $\bot \lhd \exists x.\top \cong \bot \lhd \overline{\omega}$, thus the only program satisfying S are those that behave as the identity on `nat -> nat`, such as λ `x . x`. Note that λ `x . ifzero x then x else x` does *not* satisfy this specification, as it interrogates its argument twice. If we wish to describe an identity function that can interrogate its argument an arbitrary number of times but must return the result of the first interrogation, we can use the formula $\bot \lhd \top \oslash \forall x.\bot \lhd ?(\top \oslash \omega) \lhd \exists y.y = x$ and an appropriate embedding.

- **Inflationary function:** Assuming that our language contains the constant $\leqslant$, we can define a specification on `nat -> nat` satisfied by functions that interrogate their argument once and output a value that is no smaller than the input value. This is given by $\bot \lhd \top \oslash \forall x.\bot \lhd \exists y.y \leqslant x$ and the embedding is a simple modification of the identity example above. Generalising this, we can represent arbitrary relationships between ground values, providing those relationships appear in the language $\mathcal{L}$.
- **Addition:** We can similarly define a specification on `nat -> nat -> nat` satisfied only by those functions that interrogate their arguments once (but in either order) and return the sum of their arguments. Let $S = \bot \lhd \top \oslash \forall x.\bot \lhd \top \oslash \forall y.\bot \lhd \exists z.z = x + y$. Then there are two embeddings of this formula into `nat -> nat -> nat` corresponding to whether we insist that the first or second argument is interrogated first. The formula

  $$\bot \lhd (\top \oslash \forall x.\bot \lhd \top \oslash \forall y.\bot \lhd \exists z.z = x+y) \oplus (\top \oslash \forall y.\bot \lhd \top \oslash \forall x.\bot \lhd \exists z.z = x+y)$$

  can be used to allow either order of interrogation. We could also weaken the specification further to allow multiple interrogation of arguments, as above.
- **Higher-order Functions**: We can also give specifications on higher-order types. For example, we can give a specification on `(nat -> com) -> com` which insists that the argument is only called with input 42. This can be given using the formula $(\bot \lhd \top) \lhd ?((\top \oslash \bot) \oslash !(\bot \lhd \exists x.x = 42))$ with the embedding overleaf.

  For another example, we can consider a property on programs of type `(nat -> nat) -> nat` which holds if the program behaves like λ `f . f(5) + f(6)`. This can be given using the formula

  $$\bot \lhd \top \oslash !(\bot \lhd \exists x.x = 5) \oslash \forall y.\bot \lhd \top \oslash !(\bot \lhd \exists x.x = 6) \oslash \forall z.\bot \lhd \exists w.w = y + z.$$

$$\dfrac{}{\{x\}, x \neq 42 \vdash \top}$$

$$\mathsf{P}^{x}_{\exists}\ \dfrac{\{x\}, x \neq 42 \vdash \top}{\{x\}, x \neq 42 \vdash \exists y.\top}$$

$$\dfrac{\{x\}, x \neq 42 \vdash \exists y.\top}{\{x\}, x \neq 42 \vdash \bot, \exists y.\top}$$

$$\dfrac{\{x\}, x \neq 42 \vdash \bot, \exists y.\top}{\{x\} \vdash x \neq 42, \exists y.\top}$$

$$\dfrac{\{x\} \vdash x \neq 42, \exists y.\top}{\vdash \forall x.x \neq 42, \exists y.\top}$$

$$\mathsf{P}_{\mathsf{cut}}\ \dfrac{\vdash \forall x.x \neq 42, \exists y.\top \qquad p_{\forall\omega} \vdash \forall y.\top, \omega}{\vdash \forall x.x \neq 42, \omega}$$

$$\dfrac{\vdash \forall x.x \neq 42, \omega}{\vdash \top, \forall x.x \neq 42, \omega}$$

$$\dfrac{\vdash \top, \forall x.x \neq 42, \omega}{\vdash \top \oslash \forall x.x \neq 42, \omega}$$

$$\dfrac{\vdash \top \oslash \forall x.x \neq 42, \omega}{\vdash \bot, \omega, \top \oslash \forall x.x \neq 42}$$

$$\dfrac{\vdash \bot, \omega, \top \oslash \forall x.x \neq 42}{\vdash \bot \lhd \omega, \top \oslash \forall x.x \neq 42}$$

$$\mathsf{P}^{?}_{\mathsf{der}}\ \dfrac{\vdash \bot \lhd \omega, \top \oslash \forall x.x \neq 42}{\vdash \bot \lhd \omega, ?(\top \oslash \forall x.x \neq 42)}$$

$$\mathsf{P}_{\mathsf{prom}}\ \dfrac{\vdash \bot \lhd \omega, ?(\top \oslash \forall x.x \neq 42)}{\vdash !(\bot \lhd \omega), ?(\top \oslash \forall x.x \neq 42)}$$

$$\mathsf{P}_{\multimap\mathsf{id}}\ \dfrac{\vdash !(\bot \lhd \omega), ?(\top \oslash \forall x.x \neq 42)}{\vdash \bot \lhd \top, (\top \oslash \bot)\oslash!(\bot \lhd \omega), ?(\top \oslash \forall x.x \neq 42)}$$

$$\mathsf{P}^{+}_{\mathsf{sym}}\ \dfrac{\vdash \bot \lhd \top, (\top \oslash \bot)\oslash!(\bot \lhd \omega), ?(\top \oslash \forall x.x \neq 42)}{\vdash \bot \lhd \top, ?(\top \oslash \forall x.x \neq 42), (\top \oslash \bot)\oslash!(\bot \lhd \omega)}$$

$$\dfrac{\vdash \bot \lhd \top, ?(\top \oslash \forall x.x \neq 42), (\top \oslash \bot)\oslash!(\bot \lhd \omega)}{\vdash (\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42), (\top \oslash \bot)\oslash!(\bot \lhd \omega)}$$

$$\mathsf{P}^{?}_{\mathsf{der}}\ \dfrac{\vdash (\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42), (\top \oslash \bot)\oslash!(\bot \lhd \omega)}{\vdash (\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42), ?((\top \oslash \bot)\oslash!(\bot \lhd \omega))}$$

$$\mathsf{P}_{\mathsf{prom}}\ \dfrac{\vdash (\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42), ?((\top \oslash \bot)\oslash!(\bot \lhd \omega))}{\vdash !((\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42)), ?((\top \oslash \bot)\oslash!(\bot \lhd \omega))}$$

$$\mathsf{P}_{\multimap\mathsf{id}}\ \dfrac{\vdash !((\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42)), ?((\top \oslash \bot)\oslash!(\bot \lhd \omega))}{\vdash \bot \lhd \top, (\top \oslash \bot)\oslash!((\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42)), ?((\top \oslash \bot)\oslash!(\bot \lhd \omega))}$$

$$\mathsf{P}^{+}_{\mathsf{sym}}\ \dfrac{\vdash \bot \lhd \top, (\top \oslash \bot)\oslash!((\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42)), ?((\top \oslash \bot)\oslash!(\bot \lhd \omega))}{\vdash \bot \lhd \top, ?((\top \oslash \bot)\oslash!(\bot \lhd \omega)), (\top \oslash \bot)\oslash!((\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42))}$$

$$\dfrac{\vdash \bot \lhd \top, ?((\top \oslash \bot)\oslash!(\bot \lhd \omega)), (\top \oslash \bot)\oslash!((\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42))}{\vdash (\bot \lhd \top)\lhd?((\top \oslash \bot)\oslash!(\bot \lhd \omega)), (\top \oslash \bot)\oslash!((\bot \lhd \top)\lhd?(\top \oslash \forall x.x \neq 42))}$$

- **Properties Regarding the Order of Moves**: We have seen in Section 4.1.4 that we can use nullary predicates together with the uniformity of the underlying strategies to control the order in which moves must be played. Our single-read 'good variable' property was an example of this. For another, we consider an object `(com -> nat -> com) -> com` with a `switch` method and a `read` method. We can require that if the `read` method returns 0 then the `switch` method has previously been invoked.

  This can be achieved by the embedding/specification

$$\begin{gathered}(\bot \lhd \top)\lhd?(((\top \oslash \bot)\oslash!(\bot \lhd \overline{\alpha} \oplus \overline{\omega}))\oslash!(\alpha \lhd \top)) \\ \rightharpoonup \\ (\bot \lhd \top)\lhd?(((\top \oslash \bot)\oslash!(\bot \lhd \overline{\omega}))\oslash!(\bot \lhd \top)) \\ = i((\mathtt{com} \rightarrow \mathtt{nat} \rightarrow \mathtt{com}) \rightarrow \mathtt{com})\end{gathered}$$

  using embeddings $\alpha \rightharpoonup \bot$ and $\overline{\alpha}\oplus\overline{\omega} \rightharpoonup \top\oplus\overline{\omega} \cong \overline{\omega}$. This embedding is not definable as a proof, as it is not total.

Thus, we can express a large collection of behavioural properties on TotLang programs using $\mathsf{WS1}$.

5.6 Data-independent Algorithms

We can use the fact that WS1 is a general first-order logic in a quite different way: to model *data-independent* programs. We introduced this idea in Section 4.1.4, and we expand on the theme here.

We extend TotLang with a set of atomic ground types $\mathcal{A}$: each $\phi \in \mathcal{A}$ represents an opaque set V_ϕ. For each atomic ground type ϕ, we introduce a type of ϕ-storage cells $\texttt{var}_\phi$. Otherwise, the only operation available on terms of atomic type is equality testing.

We can identify $\mathcal{A}$ with the first order language containing a unary predicate for each atomic ground type, and consider WSN over this language. The data type ϕ can be interpreted by the formula $\bot \lhd \exists x.\overline{\phi}(x)$. We can translate programs to proofs in WS1, and via this translation obtain their semantics as uniform winning strategies.

5.6.1 Programming Language

We extend TotLang with some new types and constants. For each unary predicate $\phi \in \mathcal{L}$, we introduce a ground type ϕ and type $\texttt{var}_\phi$ of storage cells of type ϕ. We also introduce the following constants:

- All constants whose type is quantified over ground types `G` are extended to include `G = `ϕ
- ϕ-variables: $\texttt{newvar}_\phi$ `:` ϕ `->` $(\texttt{var}_\phi$ `->` `G)` `->` `G`, $:=_\phi$ `:` $\texttt{var}_\phi$ `->` ϕ `->` `com`, $!_\phi$ `:` $\texttt{var}_\phi$ `->` ϕ, $\texttt{mkvar}_\phi$ `:` ϕ `->` $(\phi$ `->` `com)` `->` $\texttt{var}_\phi$
- Equality testing: `eq :` ϕ `->` ϕ `->` `nat` where `eq a b` returns `0` if `a = b` and `1` otherwise.

In this setting, the `newstack` operator from Section 5.4.1 can be generalised to a stack of ϕ values. Using this, `newset` can be typed as a function $\texttt{newset}_\phi$ `:` $((\phi$ `->` `com)` `->` $(\phi$ `->` `com)` `->` $(\phi$ `->` `nat)` `->` `nat` `->` `G)` `->` `G` which constructs a new set of whose elements have type ϕ.

5.6.2 Embedding into WSN

We can embed this language into WSN, extending the embedding in Section 5.4. We set $\mathsf{tlws}(\phi) = \bot \lhd \exists x.\overline{\phi}(x)$ and $\mathsf{tlws}(\texttt{var}_\phi) = (\bot \lhd \exists x.\phi(x)) \& (\forall x.\phi(x) \lhd \top)$.

Translation of Constants

- Equality ϕ `->` ϕ `->` `nat` is a simple modification of the proof in Section 4.1.4 replacing $\forall x.\bot$ by $\forall x.\phi(x)$.

- Sequencing, conditional, `newvar` and `encaps` extend to accommodate ϕ as an additional ground type, using the translation in Section 5.4 with G ranging over $\bot$, ω and $\forall x.\phi(x)$.
- The translation of `:=` and `!` for ϕ-variables follow that of `:=` and `!` for `nat` variables, with ω replaced by $\forall x.\bot$.
- $\texttt{newvar}_\phi$ `:` ϕ `->` $(\texttt{var}_\phi$ `->` `G)` `->` `G` is translated as follows:

$$\dfrac{\mathsf{P}_{\multimap}\dfrac{\mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash G, G^{\bot}} \quad \mathsf{P}_{\mathsf{ana}}\dfrac{p_r \qquad p_w}{\vdash \bot \lhd \exists x.\overline{\phi}(x) \& \forall x.\phi(x) \lhd \top, \bot \lhd \exists x.\overline{\phi}(x), \top \oslash \forall x.\phi(x)}}{\vdash !\mathbf{var}, \top \oslash \forall x.\phi(x)}}{\vdash G, G^{\bot} \oslash !\mathbf{var}, \top \oslash \forall x.\phi(x)}$$

where G ranges over the formula representation of ground types, p_r and p_w are:

$$\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{}{\{x\}; \overline{\phi}(x) \vdash \top}}{\{x\}; \overline{\phi}(x) \vdash \overline{\phi}(x)}}{\{x\}; \overline{\phi}(x) \vdash \exists x.\overline{\phi}(x)}}{\{x\}; \overline{\phi}(x) \vdash \bot, \exists x.\overline{\phi}(x)}}{\{x\}; \overline{\phi}(x) \vdash \bot \lhd \exists x.\overline{\phi}(x)}}{\{x\}; \overline{\phi}(x) \vdash \top, (\bot \lhd \exists x.\overline{\phi}(x))}}{\{x\}; \overline{\phi}(x) \vdash \overline{\phi}(x), (\bot \lhd \exists x.\overline{\phi}(x))}}{\{x\}; \overline{\phi}(x) \vdash \exists x.\overline{\phi}(x), (\bot \lhd \exists x.\overline{\phi}(x))}}{\{x\}; \overline{\phi}(x) \vdash \exists x.\overline{\phi}(x) \oslash (\bot \lhd \exists x.\overline{\phi}(x))}}{\{x\}; \overline{\phi}(x) \vdash \bot, \exists x.\overline{\phi}(x) \oslash (\bot \lhd \exists x.\overline{\phi}(x))}}{\{x\} \vdash \phi(x), \exists x.\overline{\phi}(x) \oslash (\bot \lhd \exists x.\overline{\phi}(x))}}{\vdash \forall x.\phi(x), \exists x.\overline{\phi}(x) \oslash (\bot \lhd \exists x.\overline{\phi}(x))}}{\vdash \top \oslash \forall x.\phi(x), \exists x.\overline{\phi}(x) \oslash (\bot \lhd \exists x.\overline{\phi}(x))}}{\vdash \bot, \exists x.\overline{\phi}(x), \bot \lhd \overline{\omega}, \top \oslash \forall x.\phi(x)}}{\vdash \bot \lhd \exists x.\overline{\phi}(x), \bot \lhd \exists x.\overline{\phi}(x), \top \oslash \forall x.\phi(x)}$$

$$\mathsf{P}^{+}_{\mathsf{wk}}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{\mathsf{P}^{x}_{\exists}\dfrac{}{\{x\}; \phi(x) \vdash \top}}{\{x\}; \phi(x) \vdash \overline{\phi}(x)}}{\{x\}; \phi(x) \vdash \exists x.\overline{\phi}(x)}}{\{x\}; \phi(x) \vdash \bot, \exists x.\overline{\phi}(x)}}{\{x\}; \phi(x) \vdash \bot \lhd \exists x.\overline{\phi}(x)}}{\{x\}; \phi(x) \vdash \top, \bot \lhd \exists x.\overline{\phi}(x)}}{\{x\}; \phi(x) \vdash \bot, \top, \bot \lhd \exists x.\overline{\phi}(x)}}{\{x\} \vdash \phi(x), \top, \bot \lhd \exists x.\overline{\phi}(x)}}{\vdash \forall x.\phi(x), \top, \bot \lhd \exists x.\overline{\phi}(x)}}{\vdash \forall x.\phi(x) \lhd \top, \bot \lhd \exists x.\overline{\phi}(x)}}{\vdash \forall x.\phi(x) \lhd \top, \bot \lhd \exists x.\overline{\phi}(x), \top \oslash \forall x.\phi(x)}$$

- `mkvar : (nat -> com) -> nat -> var` follows the TotLang embedding of `mkvar` for the `nat` type.

$$\mathsf{P}^{?}_{\mathsf{der}}\dfrac{p' \qquad \mathsf{P}^{+}_{\mathsf{wk}}\dfrac{\mathsf{P}_{\mathsf{id}}\dfrac{}{\vdash \bot \lhd \exists x.\overline{\phi}(x), \top \oslash \forall x.\phi(x)}}{\vdash \bot \lhd \exists x.\overline{\phi}(x), \top \oslash \forall x.\phi(x), (\top \oslash \bot) \oslash !(\bot \lhd \exists x.\overline{\phi}(x))}}{\vdash \bot \lhd \exists x.\overline{\phi}(x) \& \forall x.\phi(x) \lhd \top, \top \oslash \forall x.\phi(x), (\top \oslash \bot) \oslash !(\bot \lhd \exists x.\overline{\phi}(x))}}{\vdash \bot \lhd \exists x.\overline{\phi}(x) \& \forall x.\phi(x) \lhd \top, ?(\top \oslash \forall x.\phi(x)), ?((\top \oslash \bot) \oslash !(\bot \lhd \exists x.\overline{\phi}(x)))}$$

where p' is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{x\};\overline{\phi}(x)\vdash\top}}{\{x\};\overline{\phi}(x)\vdash\overline{\phi}(x)}\,\mathsf{P}^x_\exists}{\{x\};\overline{\phi}(x)\vdash\exists x.\overline{\phi}(x)}}{\{x\};\overline{\phi}(x)\vdash\bot,\exists x.\overline{\phi}(x)}}{\{x\};\overline{\phi}(x)\vdash\bot\lhd\exists x.\overline{\phi}(x)}\,\mathsf{P}_{\mathsf{prom}}}{\{x\};\overline{\phi}(x)\vdash!(\bot\lhd\exists x.\overline{\phi}(x))}}{\{x\};\overline{\phi}(x)\vdash\bot,\top}\,\mathsf{P}_{\mathsf{id}}}{\{x\};\overline{\phi}(x)\vdash\bot\otimes!(\bot\lhd\exists x.\overline{\phi}(x)),\top}\,\mathsf{P}_{\mathsf{mul}\otimes}}{\{x\};\overline{\phi}(x)\vdash\top,\bot,!(\bot\lhd\exists x.\overline{\phi}(x)),\top}}{\{x\};\overline{\phi}(x)\vdash(\top\oslash\bot)\oslash!(\bot\lhd\exists x.\overline{\phi}(x)),\top}}{\{x\};\overline{\phi}(x)\vdash\bot,\top,(\top\oslash\bot)\oslash!(\bot\lhd\exists x.\overline{\phi}(x))}}{\{x\}\vdash\phi(x),\top,(\top\oslash\bot)\oslash!(\bot\lhd\exists x.\overline{\phi}(x))}}{\vdash\forall x.\phi(x),\top,(\top\oslash\bot)\oslash!(\bot\lhd\exists x.\overline{\phi}(x))}}{\vdash\forall x.\phi(x)\lhd\top,(\top\oslash\bot)\oslash!(\bot\lhd\exists x.\overline{\phi}(x))}\,\mathsf{P}^+_{\mathsf{wk}}}{\vdash\forall x.\phi(x)\lhd\top,\top\oslash\forall x.\phi(x),(\top\oslash\bot)\oslash!(\bot\lhd\exists x.\overline{\phi}(x))}$$

Denotational Semantics

Define an *instantiation* of $\mathcal{A}$ to be a set V and family of subsets $\{V_\phi \subseteq V : \phi \in \mathcal{A}\}$. From each instantiation we define the $\mathcal{A}$-structure L_V whose underlying set is V and $I_L(\overline{\phi})(v) = \mathtt{tt}$ if and only if $v \in V_\phi$. Conversely, any such $\mathcal{A}$-structure gives rise to an instantiation.

We can give denotational semantics of this language as follows:

- Each program type maps to a sequent of WS1, and semantics of this sequent is a functor $\mathcal{M}_\emptyset \to \mathcal{G}_e$. Thus, program types can be given semantics as such functors.
- Each program $\mathtt{M} \ : \ \mathtt{T}$ maps to a WS1 proof of $\vdash \mathsf{tlws}(\mathtt{T})$, which maps to a uniform winning strategy on $I \Rightarrow [\![\mathtt{T}]\!] = [\![\mathsf{tlws}(\mathtt{T})]\!]$.

Thus, given an instantiation V and program $\mathtt{M}$ we can extract a winning strategy on $[\![\mathtt{M}]\!](L_V)$. Further, the behaviour of the resulting family of strategies is uniform with respect to the instantiated ground types.

This chapter has further demonstrated the expressivity of WS1, exhibiting a variety of stateful programs and properties upon them. This concludes the development of the WS logics of this thesis: in the next chapter we consider further directions.

Chapter 6

Further Directions

We consider some possible extensions of the work presented in this thesis.

In this thesis, we have presented a first-order logic where the computational content of a proof is stateful, together with full and faithful completeness results with respect to its simple game semantics. There are a number of further directions, both in breadth and depth, that the work can now take.

We have seen how we can use WS1 for specifying behavioural properties of imperative programs. We can seek to extend WS1 to express a larger variety of programs, and a larger variety of properties upon them.

6.1 Polymorphism

The atoms described in Chapter 4 are truly *atomic*: they range over truth and falsity, the 0- and 1-move basic games. One further direction is to consider support for propositional variables, which range over arbitrary games. For example, there are copycat morphisms on $A \otimes B \multimap B \otimes A$ and $A \otimes (A \multimap B) \multimap B$ whose underlying structure (the flow of data) can be expressed independently of the underlying games A and B (uniformly). We wish to capture such strategies as proofs. On the program side, this would allow us to represent polymorphic programs in our logic. In such a setting the following rule should be sound, when M is an arbitrary (negative) formula and X is a propositional variable:

$$\frac{\Delta \vdash \Gamma}{\Delta \vdash \Gamma[M/X]}$$

This does not hold for the atoms of WS1: contraction $\alpha \multimap \alpha \otimes \alpha$ is provable and semantically valid, but its interpretation does not scale to arbitrary games. There is a

winning strategy on $\alpha \multimap \alpha \otimes \alpha$ because our games model enforces local alternation. In the setting of [49], strategies need not be locally alternating, and one-move atoms do suffice to represent propositional variables and copycat strategies. In our setting, one approach to describing polymorphic strategies as proofs is via *move variables*, where a proof explicitly remembers moves that have been played by the environment. We sketch this approach here.

For simplicity, our starting point will be WS. The grammar of formulas will be extended with propositional variables, which come in negative (X) and positive ($\overline{X}$) pairs. The intended interpretation is that X represents an arbitrary negative game, and $\overline{X}$ denotes $X^{\perp}$. Thus sequents will be equipped with "contexts": finite sets Δ of propositional variables currently in scope. Then $\Delta \vdash \Gamma$ will be interpreted as a family of games, indexed by assignments $\Delta \to \mathcal{G}$.

We next consider core introduction rules for X and $\overline{X}$. Looking at the semantics, if Opponent is to play and his first move is in a propositional variable X, then Opponent can play any starting move in X, which denotes an unknown game. But if Opponent does play some move m, the proof term will then "know" that m is a valid opening move in X, and may play it on subsequent occasions, following [2]. Our epistemic terminology is a metaphor: we can define *uniform* strategies in this manner, in the same sense as in our first-order logic WS1. Using this approach, we must also be able to express the positive game with plays $\{s : ms \in P_X\}$ as a formula — we choose $\overline{X_m}$ (the overline is used to denote polarity). Continuing this process, we require formulas X_s and $\overline{X_s}$ for each sequence of move variables s:

$$\begin{array}{rcl} P & := & \ldots \mid \overline{X}_s \\ N & := & \ldots \mid X_s \end{array}$$

Resultantly, contexts must be enriched further, containing information such as $s : X$ where s is a sequence of move variables and X is a propositional variable. The core rules for propositional variables could be presented as follows:

$$\frac{\Gamma, sm : X, \Gamma' \vdash \perp, \overline{X_{sm}}, \Delta}{\Gamma, s : X, \Gamma' \vdash X_s, \Delta} \qquad \frac{\Gamma, sm : X, \Gamma' \vdash \top, X_{sm}, \Delta}{\Gamma, sm : X, \Gamma' \vdash \overline{X}_s, \Delta}$$

Using this approach we hope to develop a full completeness result: each finite uniform strategy is the denotation of a unique analytic proof; each uniform strategy is the denotation of a unique infinitary analytic proof.

We would also like to describe the proof denoting the identity map $\vdash X, \overline{X}$ as a finite proof using this style. However, the underlying strategy is infinite: the identity $X \multimap X$ can have an unbounded (even infinite) number of moves, depending on the size

of X. Nonetheless, it is finitely describable because it is *regular* in a certain sense. We can describe it using *looping*:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{loop(*)}}{m : X \vdash X_m, \overline{X_m}}}{m : X \vdash X_m \parr \overline{X_m}}}{m : X \vdash \top, X_m \parr \overline{X_m}}}{m : X \vdash \top, X_m, \overline{X_m}}}{m : X \vdash \overline{X}, \overline{X_m}}}{m : X \vdash \overline{X_m} \parr \overline{X}}}{m : X \vdash \bot, \overline{X_m} \parr \overline{X}}}{m : X \vdash \bot, \overline{X_m}, \overline{X}}}{* :: X \vdash X, \overline{X}}$$

Here, after the first two moves are played we can *loop* back and repeat, since the formula reached is a special case of one of its parents (the node marked with an $*$). Using this approach, we can seek to construct finite analytic proofs (with looping) of each of the non-core rules: modelling the explicit flow of data in the copycat strategies that they represent.

We can also consider the question of definability — which finitely describable but infinite-behaviour uniform strategies are definable as proofs? This can be addressed by considering the uniform strategies that are in some sense regular (the looping proofs can be related to finite automata).

6.2 Recursive Types

We have expressed an infinite stack in our logic using the exponentials, but it seems possible to give a treatment of infinite datatypes in a more principled way. In particular, we wish to introduce infrastructure for supporting *recursive types* into our logics. For example, a type of Boolean lists could be represented as $\nu X.\bot \lhd (\top \oplus (\top \oslash (\mathbf{B} \otimes X)))$. This could be useful from the programming perspective, allowing an embedding of languages with recursive types. But it is also interesting from the (purely) game semantic perspective: we have already used coinductive definitions for ! and ω, and there is no reason this technique could not be generalised. In particular, we can introduce formulas $\nu X.F(X)$ denoting the final coalgebra of the endofunctor F. Thus, $!A = \nu X.A \oslash X$ and $\omega = \nu X.\bot \& X$. The type $\nu X.A \otimes X$ would represent a random-access infinite array of A values that could be accessed in any order, where interaction in one component can affect the behaviour in other components interrogated in the future.

To do this, we would need to check that each such F *has* a final coalgebra in $\mathcal{W}$. The work in [20] shows a similar result for a category of arena games and innocent strategies, under a different set of logical operations. We conjecture that each functor made up of variables and WS type constructors has a final coalgebra, following the constructions given in the ! and ω cases. In the setting of [20], a class of definable functors also have *initial* algebras, which differ from the final ones only in their winning conditions. Perhaps there is some natural restricted class of WS expression-functors that have initial algebras, which could be denoted by corresponding catamorphism rules.

6.3 Partiality and Universality

It is known that the category $\mathcal{G}$ has a *universal object* — a game U such that for each (countable) game G there is an embedding $G \rightharpoonup U$ [51, 57]. In particular, U is the denotation of the type $\mathsf{nat} \to \mathsf{nat}$ in a stateful call-by-value language. Further, if G is the denotation of a type, then its embedding into U can be expressed as a program (in a sufficiently expressive language), which is a useful tool for proving definability and full abstraction [51,57]. This universal object can be expressed in WSN as $!(\omega \lhd \overline{\omega})$. Can the embeddings from each of the other formula objects be encoded using the rules of WS1? (i.e. just using the coalgebraic structure, without access to arbitrary recursion.)

We have also considered adding a general fixed point operator to WS!. This "logic" is inconsistent, but can be used as a low-level language for describing imperative programs with general recursion. This logic (PS) replaces the anamorphism rule by a least fixed point operator:

$$\frac{\Phi \vdash A, A^{\perp}}{\Phi \vdash A}$$

We have shown that the retractions into the universal type are definable in PS, and used this to show that each computable strategy is representable as a proof in PS.

6.4 Other Exponential Structures

Our choice of games model as a fundamental primitive has inspired our logic throughout: the rules for each connective reflect its chosen semantic interpretation. This is reasonable, because we have chosen their standard interpretation, as found in e.g. [36]. In the case of the exponential there are multiple choices: [58] identifies three linear exponential comonads on $\mathcal{G}$. As well as the sequoidal exponential studied in Chapter 3, one may also consider exponentials that allow Opponent to backtrack to any point in the play so far. There are two known variants of this — the sequential algorithms

exponential, which does not allow repetition [52], and the exponential of [31] which does. In our development we have chosen to study the sequoidal exponential due to its ability to model stateful languages and its simple algebraic properties. But one can also consider how we can extend WS with the other backtracking exponentials.

We first discuss the non-repetitive backtracking sequential algorithms exponential [13, 22, 52]. This exponential is the only one of the three that preserves finiteness of the underlying games. Thus in one sense it would have been a natural choice of exponential to model, since all formulas would represent finite games, which are only inhabited by finite strategies, and so we need only consider finite analytic proofs in our full completeness results. However, it would be inadequate for modelling state.

This exponential can be accommodated in an extension to WS. We introduce unary operators ! and ?, each of which acts on both positive and negative formulas, preserving polarity — the backtracking player need not be the starting player. The proof system exploits some key isomorphisms: in particular $!(\bot \lhd P) \cong \bot \lhd !P$. The resulting logic uses ![_] as a structural, focusing connective, so sequents are no longer just lists, but are of the form

$$S := M \mid P \mid S, M \mid S, P \mid ![S] \mid ?[S].$$

Contexts are given by

$$C\{\_\} := \_ \mid C\{\_\}, M \mid C\{\_\}, P \mid ![C\{\_\}] \mid ?[C\{\_\}]$$

and each sequent decomposes into a context and a focus. The core proof rules for WS!L (WS with the sequential algorithms exponential) are given in Figure 6-1. The rules can be given semantics using the non-repetitive backtracking exponential, and we can show that these semantics satisfy full and faithful completeness.

Expressing the *repetitive* backtracking exponential of [31] would be a pleasing goal, since it allows innocent strategies over an arena to be expressed. Speculating, one could perhaps represent both innocent and history-sensitive strategies in a combined setting, with different types representing the observational power of the strategies in question. This would add further power to the programme of using our logic for expressive typing of imperative programs — namely, sound type annotations for when they do *not* exhibit stateful behaviour.

One approach to representing this exponential in our proof system is to combine the backtracking behaviour of the sequential algorithms ! with the duplication behaviour of the sequoidal exponential — exploiting the isomorphism $!\uparrow P \cong \uparrow !P \oslash !\uparrow P$. Starting with WS!L, we replace the rules for the interaction of ! and lifts by rules such as

Figure 6-1: Core proof rules for WS!L

$$\frac{\vdash C\{(M \oslash N)\&(N \oslash M)\}}{\vdash C\{M \otimes N\}} \qquad \frac{\vdash C\{(P \lhd Q) \oplus (Q \lhd P)\}}{\vdash C\{P \parr Q\}} \qquad \frac{}{\vdash C\{\mathbf{1}\}}$$

$$\frac{\vdash C\{\bot, P \parr Q\}}{\vdash C\{\bot, P, Q\}} \qquad \frac{\vdash C\{\bot, P \oslash N)\}}{\vdash C\{\bot, P, N\}} \qquad \frac{\vdash C\{\bot, !P\}}{\vdash C\{![\bot, P]\}}$$

$$\frac{\vdash C\{\bot, ?P\}}{\vdash C\{?[\bot, P]\}} \qquad \frac{\vdash P}{\vdash \bot, P} \qquad \frac{\vdash N}{\vdash \top, N}$$

$$\frac{\vdash C\{\top, M \otimes N\}}{\vdash C\{\top, M, N\}} \qquad \frac{\vdash C\{\top, N \lhd P\}}{\vdash C\{\top, N, P\}} \qquad \frac{\vdash C\{\top, !N\}}{\vdash C\{![\top, N]\}}$$

$$\frac{\vdash C\{\top, ?N\}}{\vdash C\{?[\top, N]\}} \qquad \frac{\vdash C\{A, N\}}{\vdash C\{A \oslash N\}} \qquad \frac{\vdash C\{A, P\}}{\vdash C\{A \lhd P\}}$$

$$\frac{\vdash C\{![A]\}}{\vdash C\{!A\}} \qquad \frac{\vdash C\{?[A]\}}{\vdash C\{?A\}} \qquad \frac{\vdash C\{?M\&?N\}}{\vdash C\{?[M\&N]\}}$$

$$\frac{\vdash C\{(M \lhd P)\&(N \lhd P)\}}{\vdash C\{M\&N, P\}} \qquad \frac{\vdash C\{(M_1 \oslash N)\&(M_2 \oslash N)\}}{\vdash C\{M_1\&M_2, N\}} \qquad \frac{\vdash C\{!M \otimes !N\}}{\vdash C\{![M\&N]\}}$$

$$\frac{\vdash M \qquad \vdash N}{\vdash M\&N} \qquad \frac{\vdash C\{(P \oslash M) \oplus (Q \oslash M)\}}{\vdash C\{P \oplus Q, M\}} \qquad \frac{\vdash C\{!P \oplus !Q\}}{\vdash C\{![P \oplus Q]\}}$$

$$\frac{\vdash C\{(P_1 \lhd Q) \oplus (P_2 \lhd Q)\}}{\vdash C\{P_1 \oplus P_2, Q\}} \qquad \frac{\vdash C\{?P \parr ?Q\}}{\vdash C\{?[P \oplus Q]\}} \qquad \frac{\vdash P}{\vdash P \oplus Q}$$

$$\frac{\vdash C\{\top\}}{\vdash C\{\top, P\}} \qquad \frac{\vdash C\{\bot\}}{\vdash C\{\bot, N\}} \qquad \frac{\vdash Q}{\vdash P \oplus Q}$$

$$\frac{\vdash C\{\bot, !P, !(\bot \oslash P)\}}{\vdash C\{![\bot, P]\}}$$

We conjecture that this leads to full and faithful completeness, with each strategy representable as a unique (possibly infinitary) analytic proof.

6.5 Other Game Models

Our logic is explicitly based upon Curien-Lamarche games, for reasons we have made clear in the introduction, but one could consider WS-style logics for other games models. In particular, our strategies are deterministic, and locally alternating. One could consider a setting of Conway games, dropping the latter condition. In this case it would make sense to combine positive and negative formulas into a single class (since there are Conway games in which either player can start), and instead use two turnstiles $\vdash^+$ or $\vdash^-$ which specifies whether we are considering O-starting strategies or P-starting strategies. The rules will also need to be changed, since the linear functional extensionality isomorphism (which justifies $\mathsf{P}^{\lhd}_{\bot}$) does not hold for Conway games, but we instead have an adjunction $\mathcal{C}_s(A \multimap B, C) \cong \mathcal{C}_s(B, C \oslash A)$ [45]. To represent nondeterministic strategies, we will (at least) need another rule for $\mathsf{P}_{\parr}$ and $\mathsf{P}_{\oplus}$, representing when both branches can be taken.

In [50], a number of game models are considered, which can be differentiated abstractly by considering the algebraic properties of the sequoid. These differing algebraic properties could correspond to differing logical rules in systems such as ours.

6.6 Program Extraction

We can ask the following question: Given a logical formula M, which program types T are such that M is a (definable) specification on $\mathsf{tlws}(T)$? Given a proof $p \vdash M$, we could then compose p with the embedding, yielding a proof of $\vdash \mathsf{tlws}(T)$ that might correspond to some (imperative) program — the computational content of p. Presence of a universal type [57] in our games model (with definable retractions in a language with recursion and coroutines [51]) ensures that each computable strategy on a type object is the denotation of some program, and semantics of $\mathsf{WS1}$ proofs are computable in this sense.

As an example, the formula $A = \bot \lhd \top \oslash \forall x.\bot \lhd \exists y.(y + 1 = x \oplus x = 0)$ can be equipped with an embedding into the interpretation of the TotLang type $\mathtt{nat} \to \mathtt{nat}$ following the scheme in Section 5.5.3. Proofs of A correspond to (constructive) proofs that each element either has a predecessor, or is zero. Given any such proof, we can

extract a program on type $\texttt{nat} \to \texttt{nat}$ which computes the predecessor of a nonzero value. The result of applying the program to 0 depends on the choice of y in such cases in the proof: the proof's *computational* content.

This pattern generalises. Consider $\bot \lhd (\top \oslash \forall x_i.\bot \lhd)^n \exists y.\phi(\overrightarrow{s})$ where each s_i has free variables in $\{y, x_1, \ldots, x_n\}$. This is a specification on the program $\texttt{nat}^n$ `-> nat` and a program extracted from a proof computes for each $\overrightarrow{x_i}$ a corresponding y such that $\phi(y, x_1, \ldots, x_n)$ holds. Such formulas correspond to Π_2 formulas of the arithmetic hierarchy, and if $n = 0$ they correspond to Σ_1 formulas. Programs of type `nat -> com` correspond to Π_1 formulas. The relationship between formulas of higher arithmetic complexity and TotLang types is more subtle, due to the presence of exponentials in the translation of program types, and could be worthy of further study.

On a similar theme, we have seen that we can add peano-style axioms to WS to describe a game of natural numbers. Are there any new, stateful proofs of interesting statements of arithmetic, corresponding to imperative programs on the underlying datatypes?

Bibliography

[1] A. Abel. MiniAgda: Integrating Sized and Dependent Types. *ArXiv e-prints*, December 2010.

[2] S. Abramsky. Semantics of interaction. page 1. Springer-Verlag, 1997.

[3] S. Abramsky. Axioms for definability and full completeness. *Proof, language, and interaction: essays in honour of Robin Milner*, pages 55–75, 2000.

[4] S. Abramsky, D. R. Ghica, A. S. Murawski, and C.-H. L. Ong. Applying game semantics to compositional software modeling and verification. In *In Proceedings of TACAS 04, LNCS*, pages 421–435. Springer-Verlag, 2004.

[5] S. Abramsky, K. Honda, and G. McCusker. A fully abstract game semantics for general references. In *LICS '98: Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*, page 334, Washington, DC, USA, 1998. IEEE Computer Society.

[6] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *J. Symb. Logic*, 59(2):543–574, 1994.

[7] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full Abstraction for PCF. *Information and Computation*, 163:409–470, 1995.

[8] S. Abramsky and G. McCusker. Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions: Extended Abstract. *Electronic Notes in Theoretical Computer Science*, 3:2 – 14, 1996. Linear Logic 96 Tokyo Meeting.

[9] S. Abramsky and G. McCusker. Full abstraction for Idealized Algol with passive expressions. *Theor. Comput. Sci.*, 227(1-2):3–42, 1999.

[10] S. Abramsky and G. McCusker. Game semantics. In H. Schwichtenberg and U. Berger, editors, *Computational Logic: Proceedings of the 1997 Marktoberdorf Summer School*, pages 1–56. Springer-Verlag, 1999.

[11] The Agda proof assistant. http://wiki.portal.chalmers.se/agda/pmwiki.php.

[12] J.-M. Andreoli. Focussing and proof construction. *Annals of Pure and Applied Logic*, 107(1-3):131 – 163, 2001.

[13] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.

[14] A. Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56(1-3):183 – 220, 1992.

[15] A. Bove. *General Recursion in Type Theory*. PhD thesis, Department of Computing Science, Chalmers University of Technology, November 2002.

[16] M. Churchill and J. Laird. A logic of sequentiality. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic*, volume 6247 of *Lecture Notes in Computer Science*, pages 215–229. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-15205-4-19.

[17] M. Churchill, J. Laird, and G. McCusker. Imperative programs as proofs via game semantics. In *Logic in Computer Science (LICS), 2011 26th Annual IEEE Symposium on*, pages 65 –74, June 2011.

[18] M. Churchill and M. Takeyama. Agda implementation of game semantics and sequoidal logic. http://people.bath.ac.uk/mdc25/AgdaWS.

[19] P. Clairambault. Laird's triggering/blocking games model is equivalent to Conway games, unpublished notes. http://www.cl.cam.ac.uk/ pmc51/games_eq.pdf, 2009.

[20] P. Clairambault. Least and greatest fixpoints in game semantics. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures*, volume 5504 of *Lecture Notes in Computer Science*, pages 16–31. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-00596-1-3.

[21] J. R. B. Cockett, G. S. H. Cruttwell, and K. Saff. Combinatorial game categories. Submitted to Mathematical Structures in Computer Science, March 2010.

[22] P.-L. Curien. On the symmetry of sequentiality. In *Proceeings, 9th International Conference on Mathematical Foundations of Computer Science*, volume 802 of *LNCS*. Springer-Verlag, 1993.

[23] P.-L. Curien. Notes on game semantics, 2006.

[24] N. A. Danielsson. Beating the productivity checker using embedded languages. http://arxiv.org/abs/1012.4898.

[25] R. Davies and F. Pfenning. A modal analysis of staged computation. In *JOURNAL OF THE ACM*, pages 258–270. ACM Press, 1996.

[26] D. R. Ghica and G. McCusker. The regular-language semantics of second-order idealized algol. *Theor. Comput. Sci.*, 309:469–502, December 2003.

[27] D.R. Ghica, A.S. Murawski, and C.-H.L. Ong. Syntactic control of concurrency. *Theoretical Computer Science*, 350(2-3):234 – 251, 2006. Automata, Languages and Programming: Logic and Semantics (ICALP-B 2004).

[28] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[29] J.-Y. Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(03):301–506, 2001.

[30] T. G. Griffin. A formulae-as-types notion of control. In *In Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 47–58. ACM Press, 1990.

[31] R. Harmer, M. Hyland, and P.-A. Mellies. Categorical combinatorics for innocent strategies. In *LICS '07: Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*, pages 379–388, Washington, DC, USA, 2007. IEEE Computer Society.

[32] R. Harmer and G. McCusker. A fully abstract game semantics for finite nondeterminism. In *In Proceedings of the Fourteenth Annual Symposium on Logic in Computer Science, LICS '99. IEEE Computer*, pages 422–430. Society Press, 1999.

[33] J. Hintikka. Existential presuppositions and existential commitments. *The Journal of Philosophy*, 56(3):pp. 125–137, 1959.

[34] W. A. Howard. The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980.

[35] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000.

[36] M. Hyland. Game semantics. In P. Dybjer and A. M. Pitts, editors, *Semantics and Logics of Computation*, Publications of the Newton Institute, pages 131–184. Cambridge University Press, 1997.

[37] M. Hyland and A. Schalk. Games on graphs and sequentially realizable functionals (extended abstract). *Logic in Computer Science, Symposium on*, 0:257, 2002.

[38] G. Janelidze and G. M. Kelly. A note on actions of a monoidal category, 2001.

[39] G. Japaridze. Introduction to Computability Logic. *Annals of Pure and Applied Logic*, 123:1 – 99, 2003.

[40] J.-L. Krivine. Typed lambda-calculus in classical Zermelo-Fraenkel set theory. *ARCHIVE OF MATHEMATICAL LOGIC*, 40:2001, 2001.

[41] J.-L. Krivine. Dependent choice, 'quote' and the clock, 2002.

[42] Y. Lafont. *Logiques, categoriés et machines.* PhD thesis, Université Paris 7, 1988.

[43] J. Laird. Full abstraction for functional languages with control. In *In Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 58–67. IEEE Computer Society Press, 1997.

[44] J. Laird. A fully abstract game semantics of local exceptions. In *LICS '01: Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, page 105, Washington, DC, USA, 2001. IEEE Computer Society.

[45] J. Laird. A categorical semantics of higher order store. In *Proceedings, 9th Conference on Category Theory and Computer Science, CTCS 2002, Electronic Notes in Theoretical Computer Science.* Elsevier, 2002.

[46] J. Laird. Game semantics and Linear CPS interpretation. *Theoretical Computer Science*, 333:199–224, 2005.

[47] J. Laird. Locally Boolean domains. *Theoretical Computer Science*, 342(1):132 – 148, 2005. Applied Semantics: Selected Topics.

[48] J. Laird. A calculus of coroutines. *Theoretical Computer Science*, 350(2-3):275 – 291, 2006. Automata, Languages and Programming: Logic and Semantics (ICALP-B 2004).

[49] J. Laird. Game semantics for a polymorphic programming language. In *Proceedings of the 2010 25th Annual IEEE Symposium on Logic in Computer Science*, LICS '10, pages 41–49, Washington, DC, USA, 2010. IEEE Computer Society.

[50] J. Laird. The computational pi calclus, GaLoP workshop, 2011.

[51] J. Laird. Functional programs as coroutines : A semantic analysis. *Logical Methods in Computer Science*, to appear.

[52] F. Lamarche. Sequentiality, games and linear logic. In *Proceedings, CLICS workshop, Aarhus University.* DAIMI-397–II, 1992.

[53] O. Laurent. Polarized games. In *Logic in Computer Science, 2002. Proceedings. 17th Annual IEEE Symposium on*, pages 265–274, 2002.

[54] O. Laurent. Classical isomorphisms of types. *Mathematical. Structures in Comp. Sci.*, 15:969–1004, October 2005.

[55] O. Laurent. Game semantics for first-order logic. *Logical Methods in Computer Science*, 6(4):3, October 2010.

[56] J. Longley. Stratagem. http://homepages.inf.ed.ac.uk/jrl/stratagem/index.html.

[57] J. Longley. Universal types and what they are good for. In *Proceedings of the 2nd International Symposium on Domain Theory, Semantic Structures in Computation 3*, pages 25–63, 2003.

[58] J. Longley. Some programming languages suggested by game models. In *Proceedings of MFPS XXV*. ENTCS, 2009.

[59] J. Longley and N. Wolverson. Eriskay: a programming language based on game semantics, 2008. Games for Logic and Programming Languages III Workshop.

[60] P. Lorenzen. Ein dialogisches konstruktivitätskriterium. 1961.

[61] P. Martin-Löf. On the Meanings of the Logical Constants and the Justifications of the Logical Laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, May 1996.

[62] P.-A. Melliès, N. Tabareau, and C. Tasson. An explicit formula for the free exponential modality of linear logic. In *ICALP '09: Proceedings of the 36th Internatilonal Colloquium on Automata, Languages and Programming*, pages 247–260, Berlin, Heidelberg, 2009. Springer-Verlag.

[63] A. Murawski and N. Tzevelekos. Full abstraction for reduced ml. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures*, volume 5504 of *Lecture Notes in Computer Science*, pages 32–47. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-00596-1-4.

[64] D. Nails. Socrates. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2010 edition, 2010.

[65] H. Nickau. Hereditarily sequential functionals. In *In Proceedings of the Symposium on Logical Foundations of Computer Science: Logic at St. Petersburg, Lecture notes in Computer Science*, pages 253–264. Springer, 1994.

[66] P. W. O'Hearn and J. G. Riecke. Kripke logical relations and pcf. *Information and Computation*, 120:107–116, 1995.

[67] M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer Berlin / Heidelberg, 1992. 10.1007/BFb0013061.

[68] F. Pfenning and R. Davies. A judgmental reconstruction of modal logic. In *Mathematical Structures in Computer Science*, page 2001, 1999.

[69] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223 – 255, 1977.

[70] U. Reddy. A linear logic model of state, 1993.

[71] J. C. Reynolds. The essence of Algol. In *Proceedings of the 1981 International Symposium on Algorithmic Languages*, pages 345–372. North-Holland, 1981.

[72] A. Schalk. What is a categorical model of linear logic. Technical report, 2004.

[73] P. V. Spade. Medieval theories of obligationes. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008.

[74] V. Vene. Categorical programming with inductive and coinductive types, 2000. Diss. Math. Uni. Tartuensis, v. 23, Uni. of Tartu.

[75] N. Wolverson. *Game Semantics for an object-oriented language*. PhD thesis, University of Edinburgh, 2008.

Appendix A

Agda Formalisation

In this appendix we formalise the foundations of this work in the proof assistant Agda. This includes finitary game semantics, the syntax and semantics of WS, the full completeness procedure, and embedding of a finitary programming language. We also implement a Strategy Interaction tool allowing a user to dynamically interact with the strategy semantics of a WS proof.

Remark We only give an outline of the Agda formalisation here. The full code can be found online at [18]. The work in this appendix was created in collaboration with Makoto Takeyama.

We next describe a formalisation of game semantics and some of the work so far in the proof assistant Agda [11]. The Curry-Howard isomorphism between proofs and programs can in principle be used to write proofs in a programming language, using the type checker verify the proofs. But to express proofs of interesting propositions, *dependent types* are required. Agda is a dependently typed functional programming language, and a proof assistant, based on Martin-Löf type theory [61].

We will first show how finite games and connectives; strategies and composition can be formalised inside type theory and Agda. This is quite different to the usual set theoretic presentation, and provides some advantages and some disadvantages. We will then show how the logic WS can be embedded, give game semantics of proofs, and formalise the full completeness procedure. We will also formalise syntactic cut elimination.

We will then show how a finitary language can be embedded inside (our Agda embedding of) WS, following the ideas in Chapter 5 with bounds on variable usage. This mechanises the game semantics of such a language, by composing this embedding with the proof semantics. Finally, we will show how Agda can be used as a tool for "running" strategies in a user-friendly manner.

A.1 Game Semantics and Agda

We first formalise basic concepts of game semantics in type theory, in particular in Agda. The definitions of games, strategies, and operations on games are strikingly simple: in some sense more simple and concise then their set-theoretic counterparts. For example, we will see that implication, tensor and sequoid can be defined mutually, in a very concise manner. This is in contrast to the intricate definitions one usually sees: sequences of elements of the union of move sets, satisfying restriction properties. Composition is similarly given a simple, mechanical definition, and the fact that it type checks at all ensures immediately that prefix closure and determinacy are preserved, while in the usual set-theoretic presentation these are results that must be proved.

A.1.1 Games as Forests

The key piece of data in our definition of a CL-game is the set of plays. Given any game A, we can assume that for each move m there is a minimal play s_m such that for any play t containing m, $t \sqsupseteq s_m$. Whenever a move is considered it is always in the context of a play, and identical moves in different positions in the play forest can be replaced by distinct moves. Given such a game, we can then infer the Opponent-Player labelling by the location of a move: m is an Opponent-move if and only if it is in an odd position of s_m and $b_A = O$, or it is in an even position in s_m and $b_A = P$. Thus the data is reduced to a polarity, a set of moves, and a prefix closed set of sequences. This is just a polarity together with a forest that can branch arbitrarily.

Definitions in type theory are inductive, and an inductive definition of a forest can be given as follows:

Definition A *forest* is a pair (X,f) where X is a set and f is a family of forests, indexed by X.

We will treat f as a function from X to the collection of forests. We will call it the *children-function*, mapping a node x to its *children*, a forest. We can form the empty forest $I = (\emptyset, \mathsf{i})$ where i is the unique map whose domain is the empty set. We can construct a forest with one root node with two children as follows: $(\{q\}, \lambda q \mapsto (\{t, f\}, [t \mapsto I, f \mapsto I]))$, and so on.

We can define forests in Agda:

```
data Forest : Set1 where
  gam : {X : Set} → (X → Forest) → Forest
```

The type `Forest` has one constructor, `gam`, of the given type. The curly brackets {} indicate a *hidden argument*: the set X itself need not actually be given, because it

can be inferred from the codomain of the given f. This is an inductive definition: the resulting semantics is the initial algebra, containing forests of finite depth.

The collection of forests forms a proper class, not a set. Similarly, we have `Forest : Set1` rather than `Forest : Set`. It will be technically useful to restrict the definition of `Forest` so that it has type `Set`[1]. We can modify the definition of forest so that the collection of forests does form a set. Fix a set $\mathcal{U}$ such that each element of $\mathcal{U}$ is a set, $\mathcal{U}$ contains the empty and singleton sets, and $\mathcal{U}$ is closed under disjoint union (such a set $\mathcal{U}$ can be readily shown to exist).

Definition A *restricted forest* is a pair (X,f) where $X \in \mathcal{U}$ and f is a family of forests, indexed by X.

The collection of restricted forests does form a set — it is the initial algebra of the functor $X \mapsto (U : \mathcal{U}) \times (U \to X)$. We will henceforth assume all forests are restricted, and use the term *forest* to refer to restricted forests. Let $\mathcal{F}$ denote the set such forests.

We can define the restricted version of `Forest` in Agda. First, we define $\mathcal{U}$ via a grammar `mov` of move encodings:

```
data MovEnc : Set where
  nil : MovEnc
  one : MovEnc
  _++_ : MovEnc → MovEnc → MovEnc
```

$\mathcal{U}$ is defined to be the image of `mov` under a semantic mapping $T : \texttt{mov} \to \mathbf{Set}$. In ZF set theory, the axiom of substitution (ZF8) ensures that this is a set.

```
T : MovEnc → Set
T nil  = ⊥  -- empty set
T one  = ⊤  -- singleton set
T (x ++ y) = T x ⊎ T y  -- disjoint union
```

We can define the type of restricted forests as follows:

```
data Forest : Set where
  gam : {ι : MovEnc} → (T ι → Forest) → Forest
```

We can interpret each forest as a negative game, or as a positive game. We will use

[1] This is so that we can use standard library operations, which often defined on `Set` rather than `Set1`. Actually, in the current version of Agda this is no longer necessary as library functions are given universal definitions that work for `SetX` for arbitrary `X` (universe polymorphism). But we chose not to use this for simplicity.

```
Game : Set
Game = Forest
```

to represent games without a polarity. We define two operations on games: `Mov G` is the set of starting moves of `G`, and if `i` is such a starting move `G » i` is the resulting subgame.

```
Mov : Game → Set
Mov (gam {a} f) = T a

_»_ : (G : Game) → Mov G → Game
(gam {ι} f) ≫ i = f i
```

A.1.2 Connectives on Forests

We next give the definition of I, o, $\times$, $\otimes$, $\oslash$ and $\multimap$ on forests. These can be interpreted as operations on games as follows:

Forest	Negative Game	Positive Game
I	$\mathbf{1}$	$\mathbf{0}$
o	$\bot$	$\top$
$A \times B$	$A \& B$	$A \oplus B$
$A \otimes B$	$A \otimes B$	$A \parr B$
$A \oslash B$	$A \oslash B$	$A \lhd B$
$A \multimap B$	$A \multimap B = B \lhd A^\bot$	$B \oslash A^\bot$

Empty game I

In this case the set of root nodes is the empty set, and the children-function f is the unique map from the empty set into `Game`. This is called $\bot$-elim in the Agda standard library:

```
I : Game
I = gam {nil} ⊥-elim
```

Single-move game o

The set of root nodes (starting moves) is the singleton set $\{q\}$, denoted `one` in our Agda development. Then q has no children, so the children-function is the map $q \mapsto I$.

```
o : Game
o = gam {one} (λ _ → I)
```

Product $\times$

The set of starting moves of $A \times B$ is the disjoint union of the starting moves of A and the starting moves of B. The children of $\mathsf{inj}_1(a)$ in $A \times B$ is just the children of a in A, and the children of $\mathsf{inj}_2(b)$ in $A \times B$ is just the children of b in B. Hence we can define $\times$ as follows:

```
_×_ : Game → Game → Game
gam {i} f × gam {j} g = gam {i + j} [f, g]
```

Here $[\_, \_]$ denotes copairing.

Tensor $\otimes$, sequoid $\oslash$, implication $\multimap$

We can swiftly deal with the remaining binary operators:

```
mutual
  _⊸_ : Game → Game → Game
  G ⊸ (gam {i} f) = gam {i} (λ e → f e ⊗ G)

  _⊘_ : Game → Game → Game
  (gam {i} f) ⊘ G = gam {i} (λ e → G ⊸ f e)

  _⊗_ : Game → Game → Game
  G ⊗ H = (G ⊘ H) × (H ⊘ G)
```

We next explain these definitions. The decomposition of $\otimes$ into the product of two sequoids should be clear. For the other cases, let's view our forests as negative games.

For the definition of $\multimap$, note that the starting moves of $A \multimap B$ are the starting moves of B. Next, Player can chose to either remain in B or switch to A, and after that continue to switch back and forth at will. That is, the starting player in the children of b in $A \multimap B$ begins a play in $f(b) \otimes A$, where f is the children-function of B. Thus $X_{A \multimap B} = X_B$ and $f_{A \multimap B}(b) = f_B(b) \otimes A$.

For the definition of $\oslash$, the starting moves of $A \oslash B$ are the starting moves of A. Next, Player must play a move in A (since in $\oslash$ it is Opponent that switches). Next, Opponent may chose to play in A or B, and later switch between them. Note then that after the initial move a in $A \oslash B$ the game forest is $B \multimap f_A(a)$.

The decomposition of $\otimes$ into $\oslash$ is reflected in the $\mathsf{P}_\otimes$ rule, and the other relationships above are reflected in the core elimination rules of WS.

A.1.3 Strategies

We next define the notion of (winning) strategy on a game[2]. A *positive strategy* on a forest is a strategy on the forest for the starting player, a *negative strategy* is a strategy for the player who plays second (the *secondary player*).

A strategy for the starting player on a forest (X, f) is an element x of X together with a strategy on $f(x)$: but for the secondary player, since the starting player on (X, f) becomes the secondary player on $f(x)$. A strategy for the secondary player on (X, f) consists of, for each $x \in X$ that could be played be the starting player, a response. This response is a strategy for the starting player on $f(x)$. Thus we obtain the following mutually recursive definition of strategy, parametrised by polarity:

```
data Strat : Pol → Game → Set where
  pos : ∀ {G} → (i : Mov G) → Strat - (G ≫ i) → Strat + G
  neg : ∀ {G} → ((i : Mov G) → Strat + (G ≫ i)) → Strat - G
```

Remark We can equivalently view a `Strat - X` as a strategy for Player on the negative game whose underlying forest is `X`, and a `Strat + X` as a strategy for Player on the positive game whose underlying forest is `X`.

A.1.4 Composition

We next define composition of strategies. To compose strategies $\sigma : M \multimap N$ and $\tau : N \multimap L$, one typically considers *parallel composition plus hiding*. In Agda, we can give a mechanical definition of composition by a mutual induction. We first define the following simple procedure, taking a positive strategy on $A \otimes B$ and yielding a positive strategy on $B \otimes A$:

```
swp : ∀ {B C} → Strat + (C ⊗ B) → Strat + (B ⊗ C)
swp (pos (inj₁ x) f) = pos (inj₂ x) f
swp (pos (inj₂ y) f) = pos (inj₁ y) f
```

We can now define composition of strategies:

```
mutual
  _•₁_ : ∀ {A B C} → Strat - (A ⊸ B) → Strat - (B ⊸ C) → Strat - (A ⊸ C)
  σ •₁ (neg g) = neg (λ c → σ •₂ (g c))
  _•₂_ : ∀ {A B C} → Strat - (A ⊸ B) → Strat + (C ⊗ B) → Strat + (C ⊗ A)
```

[2]In that which follows we will leave the term "winning" implicit. Since our games here are finite, these are the total strategies.

$\sigma \bullet_2 (\mathsf{pos}\ (\mathsf{inj}_1\ \mathsf{c})\ \mathsf{g}) \;=\; \mathsf{pos}\ (\mathsf{inj}_1\ \mathsf{c})\ (\sigma \bullet_1 \mathsf{g})$

$(\mathsf{neg}\ \mathsf{f}) \bullet_2 (\mathsf{pos}\ (\mathsf{inj}_2\ \mathsf{b})\ \mathsf{g}) \;=\; \mathsf{swp}\ \$\ \mathsf{g} \bullet_2 (\mathsf{swp}\ \$\ \mathsf{f}\ \mathsf{b})$

$\_\bullet\_ \;:\; \forall\, \{\mathsf{A}\ \mathsf{B}\ \mathsf{C}\} \to \mathsf{Strat}\text{ - }(\mathsf{B} \multimap \mathsf{C}) \to \mathsf{Strat}\text{ - }(\mathsf{A} \multimap \mathsf{B}) \to \mathsf{Strat}\text{ - }(\mathsf{A} \multimap \mathsf{C})$

$\sigma \bullet \tau \;=\; \tau \bullet_1 \sigma$

The dollar operation is just right-associative application. In the above definitions we have removed some hidden arguments that have had to be manually specified, for readability (and we will continue to do so, the full version can be found in [18]).

We next give some intuition behind the above definitions. To give a strategy on $A \multimap C$ we must give a positive strategy on $A \otimes C(c)$ for each $c \in C$. But given such a c, we can provide it as an input to τ which gives us a positive winning strategy on $B \otimes C(c)$. We have hence reduced the problem to composing a negative strategy on $A \multimap B$ and a positive strategy on $B \otimes C$ to yielding a positive strategy on $A \otimes C$.

For the second procedure, suppose our second argument begins with a move c in C. Then we are provided with a negative strategy on $B \multimap C(c)$. Then we can output c in our positive strategy result, and so we only need to provide a negative strategy on $A \multimap C(c)$. This can be obtained by composing our first argument with the aforementioned negative strategy we have obtained from our second argument.

If the second argument begins with a move in b, then we are given a negative strategy on $C \multimap B(b)$. We can input this move to our first argument, yielding a positive strategy on $A \otimes B(b)$, which we can `swp` to obtain a positive strategy on $B(b) \otimes A$. We can then use our second composition procedure to compose this with the negative strategy on $C \multimap B(b)$, yielding a positive strategy on $C \otimes A$. Finally we can `swp` this to give a positive strategy on $A \otimes C$.

This corresponds to the standard definition of composition: the second procedure corresponds to when it is Player's turn to play a move, and he has a choice between two of the three components of the interaction sequence. We have otherwise just identified the symmetry between A and C.

This definition is more concise than the set-theoretic one, and immediately guarantees well-definedness. On the other hand, it does require more explanation. It corresponds explicitly to the "token pushing" mechanics of strategy composition as described by Curien [23]. '

Remark This definition of composition is related to syntactic cut elimination for WS given in Section 2.5.1. In particular, $\bullet_2$ corresponds to cut_2 and `swp` to P⅋sym.

A.1.5 Isomorphisms

We next define game isomorphisms in Agda. An isomorphism between games A and B will be a pair of strategies $A \multimap B$ and $B \multimap A$ of zig-zag shape: we will not (for our purposes) include a proof that the two strategies in question are inverses. To recall, a strategy on $A \multimap B$ is *zig-zag* if Player always switches components: the response to each O-move in A (resp. B) is a P-move in B (resp. A) [54]. Concretely, it is a strategy on the following game:

```
_⊸c_ : Game → Game → Game
G ⊸c (gam {i} f) = gam {i} (λ e → G ⊘c f e)
  where _⊘c_ : Game → Game → Game
    (gam {i} f) ⊘c G = gam {i} (λ e → G ⊸c f e)
```

It is more convenient to introduce a new data type that is equivalent to

```
Strat - (A ⊸c B)
```

rather than dealing with strategies on this game directly. We write $A \preccurlyeq B$ for this type. An isomorphism $A \approx B$ is defined to be a pair $A \preccurlyeq B$ and $B \preccurlyeq A$.

```
data _≲_ : Game → Game → Set where
  sim : ∀ {A B} →
    (h : Mov B → Mov A) →
    ((e : Mov B) → (B ≫ e) ≲ (A ≫ (h e))) →
    A ≲ B
```

This is similar to a morphism of paths in the underlying games, apart from the fact that the direction of the morphism is switched at each level of the forest to account for Opponent-Player duality. We can show in Agda that $\preccurlyeq$ is reflexive and transitive:

```
id≲ : ∀ {G} → G ≲ G
id≲ {gam f} = sim id (λ i → id≲ {f i})
_≲∘≲_ : ∀ {A B C} → B ≲ C → A ≲ B → A ≲ C
(sim f f') ≲∘≲ (sim g g') = sim (g ∘ f) (λ x → g' (f x) ≲∘≲ (f' x))
```

We can define an isomorphism to be the symmetric closure:

```
data _≈_ : Game → Game → Set where
  bsm : ∀ {A B} → B ≲ A → A ≲ B → A ≈ B
_^-1 : ∀ {M N} → M ≈ N → N ≈ M
(bsm f g) ^-1 = bsm g f
```

For example, we can give an isomorphism $M \otimes N \cong N \otimes M$:

```
sym⊗ : ∀ {M N} → (M ⊗ N) ≈ (N ⊗ M)
sym⊗ {gam f} {gam g}
  = bsm (sim [inj₂, inj₁] [(λ _ → id≲), (λ _ → id≲)])
    (sim [inj₂, inj₁] [(λ _ → id≲), (λ _ → id≲)])
```

We next show how we can convert an isomorphism to a copycat strategy, if needed:

```
copycat : ∀ {A B} → A ≲ B → Strat - (A ⊸ B)
copycat (sim f g) = neg (λ b → cc (f b) (g b))
  where cc : ∀ {A B} → (m : Mov A) → B ≲ (A ≫ m) → Strat + (B ⊗ A)
    cc m p = pos (inj₂ m) (copycat p)
```

We can use this to give an operation which composes a strategy with an isomorphism.

A.1.6 Categorical Structure

We can also formalise the structure needed to show that $\mathcal{G}_f$ is a WS-category. The operations on games are described as above; we can define the action of $\otimes$ and $\multimap$ on morphisms in a combinatorial manner:

```
mutual
  _⊗s_ : ∀ {A B C D} → Strat - (A ⊸ B) → Strat - (C ⊸ D)
    → Strat - ((A ⊗ C) ⊸ (B ⊗ D))
  (neg f) ⊗s (neg g)
    = neg [(λ x → f x ⊗s₁ (neg g)),
      (λ x → (g x ⊗s₁ (neg f)) ○≈ (_ ⊗≈ sym⊗))]
  _⊗s₁_ : ∀ {A B C D} → Strat + (B ⊗ A) → Strat - (C ⊸ D)
    → Strat + ((D ⊸ B) ⊗ (A ⊗ C))
  (pos (inj₁ x) f) ⊗s₁ σ = pos (inj₁ x) (f ⊗s σ)
  (pos (inj₂ y) f) ⊗s₁ σ = pos (inj₂ (inj₁ y)) (σ ⊸s f)
  _⊸s_ : ∀ {A B C D} → Strat - (C ⊸ A) → Strat - (B ⊸ D)
    → Strat - ((A ⊸ B) ⊸ (C ⊸ D))
  σ ⊸s (neg g) = neg (λ x → σ ⊸s₁ (g x))
  _⊸s₁_ : ∀ {A B C D} → Strat - (C ⊸ A) → Strat + (D ⊗ B)
    → Strat + ((D ⊗ C) ⊗ (A ⊸ B))
  σ ⊸s₁ τ = ((τ ○≈ sym⊗) ⊗s₁ σ) ○≈ sym⊗
```

where:

```
_⊗≈_ : ∀ {M N} O → M ≈ N → (O ⊗ M) ≈ (O ⊗ N)
```

The product structure is simple. For the sequoid we have to define strictness. A strict strategy on $A \multimap B$ is a strategy on $A \hat{\multimap} B$ where:

```
_—∘^_ : Game → Game → Game
G —∘^ (gam {i} f) = gam {i} (λ e → G ⊘ f e)
```

We also need to check that the identity is strict and composition preserves strictness, which can be achieved by refining the type of `copycat` above and defining strict form of composition.

```
copycat_st : ∀ {A B} → A ≲ B → Strat - (A —∘^ B)
copycat_st (sim f g) = neg (λ b → cc (f b) (g b))
  where cc : ∀ {A B} → (m : Mov A) → B ≲ (A ≫ m) → Strat + (A ⊘ B)
    cc m p = pos m (copycat p)
_•^_ : ∀ {A B C} → Strat - (B —∘^ C) → Strat - (A —∘^ B) → Strat - (A —∘^ C)
```

We can then define the action of $\oslash$ on (strict) strategies.

We can define isomorphisms such as associativity of $\otimes$, the action isomorphisms for $\oslash$, decomposability, linear functional extensionality etc. We give some examples:

```
lfe : ∀ {M N} → (M —∘ N) —∘o ≈ (N —∘o) ⊘ M
lfe = bsm (sim id (λ _ → id≲)) (sim id (λ _ → id≲))
mutual
  pasc—∘ : ∀ {M N O} → M —∘ (N —∘ O) ≈ (M ⊗ N) —∘ O
  pasc—∘ = bsm (sim id (λ i → (_ ⊗≲ (wk₂ $ sym⊗)) ≲∘≲ (wk₁ $ asc⊗)))
               (sim id (λ i → (wk₂ $ asc⊗) ≲∘≲ (_ ⊗≲ (wk₁ $ sym⊗))))
  asc⊗ : ∀ {M N O} → (M ⊗ N) ⊗ O ≈ M ⊗ (N ⊗ O)
  asc⊗ = bsm (sim [[inj₁, inj₂ ∘ inj₁], inj₂ ∘ inj₂]
                  [[(λ i → (wk₂ (sym⊗) ≲—∘ _) ≲∘≲ (wk₁ $ pasc—∘))
                    , (λ i → wk₂ $ psym—∘)], (λ i → wk₂ $ pasc—∘)])
             (sim [inj₁ ∘ inj₁, [inj₁ ∘ inj₂, inj₂]]
                  [(λ i → (wk₂ $ pasc—∘) ≲∘≲ (wk₁ (sym⊗) ≲—∘ _))
                  , [(λ i → wk₂ $ psym—∘), (λ i → wk₁ $ pasc—∘)]])
  psym—∘ : ∀ {M N O} → M —∘ (N —∘ O) ≈ N —∘ (M —∘ O)
  psym—∘ = pasc—∘ ≈∘≈ (sym⊗ ≈—∘ _) ≈∘≈ (pasc—∘ ^-1)
pasc⊘ : ∀ {M N O} → (M ⊘ N) ⊘ O ≈ M ⊘ (N ⊗ O)
pasc⊘ = bsm (sim id (λ i → (wk₂ (sym⊗) ≲—∘ (m i)) ≲∘≲ (wk₁ $ pasc—∘)))
```

```
  (sim id (λ i → (wk₂ $ pasc⊸) ≲∘≲ (wk₂ (sym⊗) ≲⊸ _)))
psym⊘ : ∀ {M N O} → (M ⊘ N) ⊘ O ≈ (M ⊘ O) ⊘ N
psym⊘ = pasc⊘ ≈∘≈ (_ ⊘≈ sym⊗) ≈∘≈ (pasc⊘ ^-1)
```

In the final definition, we see how infix notation and unicode allows us to write Agda definitions that look like the categorical semantics. We can use the machinery above to give game semantics WS proofs.

A.2 WS and Agda

A.2.1 Formulas and Proofs

It is straightforward to formalise the syntax of WS in Agda. We define an inductive type for formulas, parametrised on polarity; and an inductive type for proofs, parametrised on the sequent that they are proving.

```
data Fml : Pol → Set where
  F0  : Fml +
  F1  : Fml -
  F⊥  : Fml -
  F⊤  : Fml +
  _⊗_  : (M N : Fml -) → Fml -
  _par_  : (P Q : Fml +) → Fml +
  _⊕_  : (P Q : Fml +) → Fml +
  _&_  : (M N : Fml -) → Fml -
  _⊘_  : ∀ {p} → (A : Fml p) (M : Fml -) → Fml p
  _<|_  : ∀ {p} → (A : Fml p) (P : Fml +) → Fml p
_⊥' : {p : Pol} → Fml p → Fml (¬ p)
F0 ⊥' = F1
...
data Ctx : Set where
  ε   : Ctx
  _,_ : ∀ {p} → Fml p → Ctx → Ctx
[_] : ∀ {p} → Fml p → Ctx
[A] = A,ε
_,,_ : Ctx → Ctx → Ctx
data Seq (p : Pol) : Set where
  _,_ : Fml p → Ctx → Seq p
```

```
[_] s : ∀ {p} → Fml p → Seq p
_,,0_ : ∀ {p} → Seq p → Ctx → Seq p
...
  -- Predicate: ctxpol p Δ holds if all elements in Δ have polarity p.
data ctxpol (p : Pol) : Ctx → Set where
  ε   : ctxpol p ε
  _,_ : ∀ {Γ} → (P : Fml p) → ctxpol p Γ → ctxpol p (P, Γ)
  -- Proof rules of WS
data ⊢_ : ∀ {p} → Seq p → Set where
  P1  : ∀ {Γ} → ⊢ F1, Γ
  P⊤  : ⊢ F⊤, ε
  P⊗  : ∀ {M N Γ} → ⊢ M, N, Γ → ⊢ N, M, Γ → ⊢ M ⊗ N, Γ
  P&  : ∀ {M N Γ} → ⊢ M, Γ → ⊢ N, Γ → ⊢ M & N, Γ
  P⊥+ : ∀ {P : Fml +} → ⊢ P, ε → ⊢ F⊥, P, ε
  P⊥- : ∀ {N : Fml -} {Γ} → ⊢ F⊥, Γ → ⊢ F⊥, N, Γ
  P⊥⊘ : ∀ {P : Fml +} {N Γ} → ⊢ F⊥, P ⊘ N, Γ → ⊢ F⊥, P, N, Γ
  ...
  P⊗T : ∀ {p M N Δ} {Γ : Seq p} → ⊢ Γ,,0 M, N, Δ → ⊢ Γ,,0 M ⊗ N, Δ
  PparT : ∀ {p P Q Δ} {Γ : Seq p} → ⊢ Γ,,0 P, Q, Δ → ⊢ Γ,,0 P par Q, Δ
  P⊕T1 : ∀ {p P Q Δ} {Γ : Seq p} → ⊢ Γ,,0 P, Δ → ⊢ Γ,,0 P ⊕ Q, Δ
  Pwk  : ∀ {p Δ} {Γ : Seq p} {M : Fml -} → ⊢ Γ,,0 M, Δ → ⊢ Γ,,0 Δ
  Pcut : ∀ {p Δ Γ1} {Γ : Seq p} {M : Fml -} → ctxpol + Δ →
    ⊢ Γ,,0 M ⊥', Γ1 → ⊢ M, Δ → ⊢ Γ,,0 Δ,,0 Γ1
  ...
```

A.2.2 Semantics of Sequents

Semantics of Formulas

We can give semantics of formulas and sequents as games.

```
⟦_⟧ : {p : Pol} → Fml p → Game
⟦ F1 ⟧     = I
⟦ F⊥ ⟧     = o
⟦ F0 ⟧     = I
⟦ F⊤ ⟧     = o
⟦ M ⊗ N ⟧ = ⟦ M ⟧ ⊗' ⟦ N ⟧
```

```
⟦ P par Q ⟧ = ⟦ P ⟧ ⊗' ⟦ Q ⟧
⟦ M & N ⟧ = ⟦ M ⟧ ×' ⟦ N ⟧
⟦ P ⊕ Q ⟧ = ⟦ P ⟧ ×' ⟦ Q ⟧
⟦ _⊘_ {-} M N ⟧ = ⟦ M ⟧ ⊘' ⟦ N ⟧
⟦ _⊘_ {+} P N ⟧ = ⟦ N ⟧ ⊸ ⟦ P ⟧
⟦ _<|_ {+} P Q ⟧ = ⟦ P ⟧ ⊘' ⟦ Q ⟧
⟦ _<|_ {-} M P ⟧ = ⟦ P ⟧ ⊸ ⟦ M ⟧

_F : ∀ {p} → Seq p → Fml p
_F (A, ε) = A
_F (A, (_, _ {-} M Γ)) = (A ⊘ M, Γ) F
_F (A, (_, _ {+} P Γ)) = (A <| P, Γ) F

⟦_⟧' : ∀ {p} → Seq p → Game
⟦ Γ ⟧' = ⟦ Γ F ⟧
```

The operators on the right hand side of the first definition are the semantic operations on games defined above, temporarily primed in the WS-semantics module for disambiguation.

Semantics of Contexts

In order to give semantics of proof rules, we must define semantics of contexts as functors on strict strategies and isomorphisms.

```
⟦_⟧- : Ctx → Game → Game
⟦ ε ⟧- G = G
⟦ _, _ {-} M Γ ⟧- G = ⟦ Γ ⟧- (G ⊘' ⟦ M ⟧)
⟦ _, _ {+} P Γ ⟧- G = ⟦ Γ ⟧- (⟦ P ⟧ ⊸ G)

⟦_⟧≲- : ∀ {M N} Γ → M ≲ N → ⟦ Γ ⟧- M ≲ ⟦ Γ ⟧- N
⟦ ε ⟧≲- c = c
⟦ _, _ {+} P Γ ⟧≲- c = ⟦ Γ ⟧≲- (⟦ P ⟧ ⊸≲ c)
⟦ _, _ {-} O Γ ⟧≲- c = ⟦ Γ ⟧≲- (c ≲⊘ ⟦ O ⟧)

⟦_⟧+ : Ctx → Game → Game
⟦_⟧≲+ : ∀ {M N} Γ → M ≲ N → (⟦ Γ ⟧+ M) ≲ (⟦ Γ ⟧+ N)
⟦_⟧≈- : ∀ {M N} Γ → M ≈ N → (⟦ Γ ⟧- M) ≈ (⟦ Γ ⟧- N)
⟦_⟧≈+ : ∀ {M N} Γ → M ≈ N → (⟦ Γ ⟧+ M) ≈ (⟦ Γ ⟧+ N)
⟦_⟧σ- : ∀ {M N} Γ → Strat - (M ⊸^ N) → Strat - (⟦ Γ ⟧- M ⊸^ ⟦ Γ ⟧- N)
⟦_⟧σ+ : ∀ {M N} Γ → Strat - (M ⊸^ N) → Strat - (⟦ Γ ⟧+ M ⊸^ ⟦ Γ ⟧+ N)
```

We next formalise some simple equality proofs in Agda.

```
⟦Δ⟧-⟦M⟧≡⟦M, Δ⟧ : ∀ {M : Fml -} Γ → ⟦ Γ ⟧- ⟦ M ⟧ ≡ ⟦ M, Γ ⟧'
⟦Δ⟧-⟦M⟧≡⟦M, Δ⟧ ε = refl
⟦Δ⟧-⟦M⟧≡⟦M, Δ⟧ (_, _ {-} _ Γ) = ⟦Δ⟧-⟦M⟧≡⟦M, Δ⟧ Γ
⟦Δ⟧-⟦M⟧≡⟦M, Δ⟧ (_, _ {+} _ Γ) = ⟦Δ⟧-⟦M⟧≡⟦M, Δ⟧ Γ
⟦Δ⟧+⟦P⟧≡⟦P, Δ⟧ : ∀ {P : Fml +} Γ → ⟦ Γ ⟧+ ⟦ P ⟧ ≡ ⟦ P, Γ ⟧'
⟦Δ⟧-≡⟦, Δ⟧ : ∀ (Γ : Seq -) Δ → ⟦ Δ ⟧- ⟦ Γ ⟧' ≡ ⟦ Γ, , 0 Δ ⟧'
⟦, Δ⟧≡⟦Δ⟧+ : ∀ (Γ : Seq +) Δ → ⟦ Γ, , 0 Δ ⟧' ≡ ⟦ Δ ⟧+ ⟦ Γ ⟧'
⟦, Δ⟧≡⟦Δ⟧- : ∀ (Γ : Seq -) Δ → ⟦ Γ, , 0 Δ ⟧' ≡ ⟦ Δ ⟧- ⟦ Γ ⟧'
```

We can also formalise distributivity isomorphisms.

```
distΓ- : ∀ Γ {M N} → ⟦ Γ ⟧- (M ×' N) ≈ ⟦ Γ ⟧- M ×' ⟦ Γ ⟧- N
distΓ- ε = id≈
distΓ- (_, _ {-} _ Γ) = (⟦ Γ ⟧≈- dist1) ≈○≈ (distΓ- Γ)
distΓ- (_, _ {+} _ Γ) = (⟦ Γ ⟧≈- dist2) ≈○≈ (distΓ- Γ)
Γ-I≈I : ∀ Γ → ⟦ Γ ⟧- I ≈ I
distΓ+ : ∀ Γ {M N} → ⟦ Γ ⟧+ (M ×' N) ≈ ⟦ Γ ⟧+ M ×' ⟦ Γ ⟧+ N
```

A.2.3 Semantics of Proofs

The structure outlined above can be used to give semantics of WS proofs:

```
⟦_⟧⊢ : ∀ {p} {Γ : Seq p} → ⊢ Γ → Strat p ⟦ Γ ⟧'
⟦_⟧⊢ {.p} (P1T {p} {Γ} {Δ} y)              = ⟦P1T⟧ {p} {Δ} {Γ} ⟦ y ⟧⊢
⟦_⟧⊢ P⊤                                    = ⟦P⊤⟧
⟦_⟧⊢ {.p} (P0T {p} {Γ} {Δ} y)              = ⟦P0T⟧ {p} {Δ} {Γ} ⟦ y ⟧⊢
⟦_⟧⊢ {.p} (P⊗T {p} {M} {N} {Δ} {Γ} y)      = ⟦P⊗T⟧ {p} {M} {N} {Δ} {Γ} ⟦ y ⟧⊢
⟦_⟧⊢ {.p} (PparT {p} {M} {N} {Δ} {Γ} y)    = ⟦PparT⟧ {p} {M} {N} {Δ} {Γ} ⟦ y ⟧⊢
...
```

Each of the remaining cases are similar, calling an auxiliary function that defines the semantics of that rule. We give some samples:

```
⟦P1⟧ : ∀ {Γ} → Strat - ⟦ F1, Γ ⟧'
⟦P1⟧ {Γ} rewrite ⟦, Δ⟧≡⟦Δ⟧- [F1] s Γ = Γ-I≈I Γ ^-1 ≈○ σI
⟦P&⟧ : ∀ {Γ M N} →
  Strat - ⟦ M, Γ ⟧' → Strat - ⟦ N, Γ ⟧' → Strat - ⟦ M & N, Γ ⟧'
⟦P&⟧ {Γ} {M} {N} σ τ
    rewrite ⟦, Δ⟧≡⟦Δ⟧- [M & N] s Γ | ⟦, Δ⟧≡⟦Δ⟧- [M] s Γ | ⟦, Δ⟧≡⟦Δ⟧- [N] s Γ
```

```
      = distΓ- Γ ^-1 ≈∘ (σ σ& τ)
⟦P⊗⟧ : ∀ {Γ M N} →
  Strat - ⟦ M, N, Γ ⟧' → Strat - ⟦ N, M, Γ ⟧' → Strat - ⟦ M ⊗ N, Γ ⟧'
⟦P⊗⟧ {Γ} {M} {N} σ τ
  rewrite ⟦, Δ⟧≡⟦Δ⟧- [M ⊗ N] s Γ | ⟦, Δ⟧≡⟦Δ⟧- (M, N, ε) Γ | ⟦, Δ⟧≡⟦Δ⟧- (N, M, ε) Γ
    = ⟦ Γ ⟧≈- (dec1 ^-1) ≈∘ distΓ- Γ ^-1 ≈∘ (σ σ& τ)
⟦P⊘⟧ : ∀ {Γ : Ctx} {p : Pol} {A : Fml p} {N : Fml -} →
  Strat p ⟦ A, N, Γ ⟧' → Strat p ⟦ A ⊘ N, Γ ⟧'
⟦P⊘⟧ σ = σ
```

The above definitions are of the form

```
⟦rule⟧ : type
⟦rule⟧ args rewrite eqns = formula
```

To read this, one can ignore the rewrite eqns part. This is only needed for Agda to be convinced that the formula type-checks. Each of the terms in eqns is a proof of $A = B$ for some A and B. The rewrite command instructs Agda to use these eqns to replace instances of A for instances of B in the goal type, allowing the term to type-check. Thus, to see that

```
distΓ- Γ ^-1 ≈∘ (σ σ& τ)
```

really does have type

```
Strat - ⟦ M & N, Γ ⟧'
```

we must use the context lemmas, which occur here as proofs such as

```
⟦, Δ⟧≡⟦Δ⟧- [M & N] s Γ
```

It is pleasing that the formula part of these equations really looks just like the categorical semantics of WS.

We can also give semantics to non-core rules. In the case of $\mathsf{P}_{\mathsf{mix}}$ we use the tensor structure on arrows, for $\mathsf{P}_{\mathsf{cut}}$ we use composition. We can thus complete the definition of the function

```
⟦_⟧⊢ : ∀ {p} {Γ : Seq p} → ⊢ Γ → Strat p ⟦ Γ ⟧'
```

which gives the semantics of a WS proof as a total strategy on the appropriate game.

A.2.4 Full Completeness

We can also formalise the full completeness procedure for WS in Agda.

```
reify : ∀ {p} {Γ : Seq p} → Strat p ⟦ Γ ⟧' → ⊢ Γ
```

To do this, we show that each of the core rules are invertible, from a semantic perspective. For example,

```
un⟦Ppar⟧ : ∀ {Γ P Q} → Strat + ⟦ P par Q, Γ ⟧'
  → Strat + ⟦ P, Q, Γ ⟧' ⊎ Strat + ⟦ Q, P, Γ ⟧'
un⟦Ppar⟧ {Γ} {P} {Q} σ
  rewrite ⟦, Δ⟧≡⟦Δ⟧+ [P par Q] s Γ | ⟦, Δ⟧≡⟦Δ⟧+ (P, Q, ε) Γ | ...
  = coprod $ σ ○≈ ⟦ Γ ⟧≈+ dec1 ○≈ distΓ+ Γ

un⟦P⊗1⟧ : ∀ {Γ M N} → Strat - ⟦ M ⊗ N, Γ ⟧' → Strat - ⟦ M, N, Γ ⟧'
un⟦P⊗1⟧ {Γ} {M} {N} σ
  rewrite ⟦, Δ⟧≡⟦Δ⟧- [M ⊗ N] s Γ | ⟦, Δ⟧≡⟦Δ⟧- (M, N, ε) Γ
  = pi1 $ distΓ- Γ ≈○ ⟦ Γ ⟧≈- dec1 ≈○ σ

un⟦P⊥⊘⟧ : ∀ {Γ} {P} {N} → Strat - ⟦ F⊥, P, N, Γ ⟧' → Strat - ⟦ F⊥, P ⊘ N, Γ ⟧'
un⟦P⊥⊘⟧ {Γ} {P} {N} σ
  rewrite ⟦, Δ⟧≡⟦Δ⟧- (F⊥, P, N, ε) Γ | ⟦, Δ⟧≡⟦Δ⟧- (F⊥, P ⊘ N, ε) Γ
  = ⟦ Γ ⟧≈- (lfe' ^-1) ≈○ σ

un⟦P&1⟧ : ∀ {Γ M N} → Strat - ⟦ M & N, Γ ⟧' → Strat - ⟦ M, Γ ⟧'
un⟦P&1⟧ {Γ} {M} {N} σ rewrite ⟦, Δ⟧≡⟦Δ⟧- [M & N] ' Γ | ⟦,Δ⟧≡⟦Δ⟧- [ M ]' Γ
  = pi1 $ distΓ- Γ ≈○ σ
```

The other main component of the full completeness result is the fact that reify terminates. In Agda, all recursive definitions must use structural induction. If we were to write reify above directly by induction following the definitions given in Figure 2-7, Agda would reject it. The termination argument in Proposition 2.4.3 used a compound lexicographical ordering, and so to convince Agda that reify is terminating, we must somehow reflect this.

Termination Checking

We can use a technique introduced by Bove [15] to partition the the full completeness procedure from its termination proof. The idea is to construct a Dom object which is defined as an inductive data type following the structure of the proof. The full completeness procedure itself can then be defined by induction on this object in a purely structural way. Showing that the procedure terminates is then reduced to constructing

a Dom object. We can think of $\mathsf{Dom}(\sigma, \Gamma)$ as an inductively defined predicate that holds if reify terminates on arguments σ,Γ.

$$
\begin{array}{l}
\textbf{data}\ \mathsf{Dom} : \forall\,\{\mathsf{p}\}\,(\Gamma : \mathsf{Seq\ p}) \to \mathsf{Set}\ \textbf{where} \\
\quad \mathsf{DF0} \quad : \forall\,\{\Gamma\} \to \mathsf{Dom}\,(\mathsf{F0}, \Gamma) \\
\quad \mathsf{DF1} \quad : \forall\,\{\Gamma\} \to \mathsf{Dom}\,(\mathsf{F1}, \Gamma) \\
\quad \mathsf{DF}\bot \quad : \mathsf{Dom}\,(\mathsf{F}\bot, \varepsilon) \\
\quad \mathsf{DF}\top \quad : \mathsf{Dom}\,(\mathsf{F}\top, \varepsilon) \\
\quad \mathsf{D}\bot\mathsf{N}\Gamma : \forall\,\{\Gamma\}\,\{\mathsf{N}\} \to \mathsf{Dom}\,(\mathsf{F}\bot, \Gamma) \to \mathsf{Dom}\,(\mathsf{F}\bot, \mathsf{N}, \Gamma) \\
\quad \mathsf{D}\bot\mathsf{P}\varepsilon : \forall\,\{\mathsf{P}\} \quad \to \mathsf{Dom}\,(\mathsf{P}, \varepsilon) \quad \to \mathsf{Dom}\,(\mathsf{F}\bot, \mathsf{P}, \varepsilon) \\
\quad \mathsf{D}\bot\mathsf{PQ}\Gamma : \forall\,\{\Gamma\ \mathsf{P}\ \mathsf{Q}\} \to \mathsf{Dom}\,(\mathsf{F}\bot, \mathsf{P}\ \mathsf{par}\ \mathsf{Q}, \Gamma) \to \mathsf{Dom}\,(\mathsf{F}\bot, \mathsf{P}, \mathsf{Q}, \Gamma) \\
\quad \mathsf{D}\bot\mathsf{PM}\Gamma : \forall\,\{\Gamma\ \mathsf{P}\ \mathsf{M}\} \to \mathsf{Dom}\,(\mathsf{F}\bot, \mathsf{P} \oslash \mathsf{M}, \Gamma) \to \mathsf{Dom}\,(\mathsf{F}\bot, \mathsf{P}, \mathsf{M}, \Gamma) \\
\quad \mathsf{D}\top\mathsf{P}\Gamma : \forall\,\{\Gamma\}\,\{\mathsf{P}\} \to \mathsf{Dom}\,(\mathsf{F}\top, \Gamma) \to \mathsf{Dom}\,(\mathsf{F}\top, \mathsf{P}, \Gamma) \\
\quad \mathsf{D}\top\mathsf{N}\varepsilon : \forall\,\{\mathsf{N}\} \quad \to \mathsf{Dom}\,(\mathsf{N}, \varepsilon) \quad \to \mathsf{Dom}\,(\mathsf{F}\top, \mathsf{N}, \varepsilon) \\
\quad \mathsf{D}\top\mathsf{NQ}\Gamma : \forall\,\{\Gamma\ \mathsf{N}\ \mathsf{Q}\} \to \mathsf{Dom}\,(\mathsf{F}\top, \mathsf{N} <\mid \mathsf{Q}, \Gamma) \to \mathsf{Dom}\,(\mathsf{F}\top, \mathsf{N}, \mathsf{Q}, \Gamma) \\
\quad \mathsf{D}\top\mathsf{NM}\Gamma : \forall\,\{\Gamma\ \mathsf{N}\ \mathsf{M}\} \to \mathsf{Dom}\,(\mathsf{F}\top, \mathsf{N} \otimes \mathsf{M}, \Gamma) \to \mathsf{Dom}\,(\mathsf{F}\top, \mathsf{N}, \mathsf{M}, \Gamma) \\
\quad \mathsf{DM}\otimes\mathsf{N}\Gamma : \forall\,\{\Gamma\ \mathsf{M}\ \mathsf{N}\} \to \mathsf{Dom}\,(\mathsf{M}, \mathsf{N}, \Gamma) \to \mathsf{Dom}\,(\mathsf{N}, \mathsf{M}, \Gamma) \to \mathsf{Dom}\,(\mathsf{M} \otimes \mathsf{N}, \Gamma) \\
\quad \mathsf{DPparQ}\Gamma : \forall\,\{\Gamma\ \mathsf{P}\ \mathsf{Q}\} \to \mathsf{Dom}\,(\mathsf{P}, \mathsf{Q}, \Gamma) \to \mathsf{Dom}\,(\mathsf{Q}, \mathsf{P}, \Gamma) \to \mathsf{Dom}\,(\mathsf{P}\ \mathsf{par}\ \mathsf{Q}, \Gamma) \\
\quad \mathsf{DP}\oplus\mathsf{Q}\Gamma : \forall\,\{\Gamma\ \mathsf{P}\ \mathsf{Q}\} \to \mathsf{Dom}\,(\mathsf{P}, \Gamma) \quad \to \mathsf{Dom}\,(\mathsf{Q}, \Gamma) \to \mathsf{Dom}\,(\mathsf{P} \oplus \mathsf{Q}, \Gamma) \\
\quad \mathsf{DM}\&\mathsf{N}\Gamma : \forall\,\{\Gamma\ \mathsf{M}\ \mathsf{N}\} \to \mathsf{Dom}\,(\mathsf{M}, \Gamma) \to \mathsf{Dom}\,(\mathsf{N}, \Gamma) \to \mathsf{Dom}\,(\mathsf{M}\ \&\ \mathsf{N}, \Gamma) \\
\quad \mathsf{DA}\oslash\mathsf{M}\Gamma : \forall\,\{\Gamma\}\,\{\mathsf{p}\}\,\{\mathsf{A} : \mathsf{Fml\ p}\}\,\{\mathsf{M}\} \to \mathsf{Dom}\,(\mathsf{A}, \mathsf{M}, \Gamma) \to \mathsf{Dom}\,(\mathsf{A} \oslash \mathsf{M}, \Gamma) \\
\quad \mathsf{DA}<\mid\mathsf{P}\Gamma : \forall\,\{\Gamma\}\,\{\mathsf{p}\}\,\{\mathsf{A} : \mathsf{Fml\ p}\}\,\{\mathsf{P}\} \to \mathsf{Dom}\,(\mathsf{A}, \mathsf{P}, \Gamma) \to \mathsf{Dom}\,(\mathsf{A} <\mid \mathsf{P}, \Gamma) \\
\mathsf{reif} : \forall\,\{\mathsf{p}\}\,\{\Gamma : \mathsf{Seq\ p}\}\,(\mathsf{h} : \mathsf{Dom}\,\Gamma) \to \mathsf{Strat\ p}\,[\![\ \Gamma\ ]\!]' \to \vdash \Gamma \\
\mathsf{reif}\ \mathsf{DF1} \qquad\qquad \sigma = \mathsf{P1} \\
\mathsf{reif}\ \mathsf{DF}\top \qquad\qquad \sigma = \mathsf{P}\top \\
\mathsf{reif}\ (\mathsf{D}\bot\mathsf{N}\Gamma\ \{\Gamma\}\ \mathsf{h}) \quad \sigma = \mathsf{P}\bot\text{-} \ \$\ \mathsf{reif\ h}\ \$\ \mathsf{un}[\![\mathsf{P}\bot\text{-}]\!]\ \{\Gamma\}\ \sigma \\
\mathsf{reif}\ (\mathsf{D}\bot\mathsf{P}\varepsilon\ \{\mathsf{P}\}\ \mathsf{h}) \quad \sigma = \mathsf{P}\bot+ \ \$\ \mathsf{reif\ h}\ \$\ \mathsf{un}[\![\mathsf{P}\bot+]\!]\ \{\mathsf{P}\}\ \sigma \\
\mathsf{reif}\ (\mathsf{D}\bot\mathsf{PQ}\Gamma\ \{\Gamma\}\ \mathsf{h})\ \sigma = \mathsf{P}\bot\mathsf{par}\ \$\ \mathsf{reif\ h}\ \$\ \mathsf{un}[\![\mathsf{P}\bot\mathsf{par}]\!]\ \{\Gamma\}\ \sigma \\
\mathsf{reif}\ (\mathsf{D}\bot\mathsf{PM}\Gamma\ \{\Gamma\}\ \mathsf{h})\ \sigma = \mathsf{P}\bot\oslash\ \$\ \mathsf{reif\ h}\ \$\ \mathsf{un}[\![\mathsf{P}\bot\oslash]\!]\ \{\Gamma\}\ \sigma \\
\mathsf{reif}\ (\mathsf{DM}\otimes\mathsf{N}\Gamma\ \{\Gamma\}\ \mathsf{h}\ \mathsf{g})\ \sigma = \mathsf{P}\otimes\ (\mathsf{reif\ h}\ \$\ \mathsf{un}[\![\mathsf{P}\otimes 1]\!]\ \{\Gamma\}\ \sigma)\ \$\ \mathsf{reif\ g}\ \$\ \mathsf{un}[\![\mathsf{P}\otimes 2]\!]\ \{\Gamma\}\ \sigma \\
\mathsf{reif}\ (\mathsf{DP}\oplus\mathsf{Q}\Gamma\ \{\Gamma\}\ \mathsf{h}\ \mathsf{g})\ \sigma = [\mathsf{P}\oplus_1 \circ \mathsf{reif\ h}, \mathsf{P}\oplus_2 \circ \mathsf{reif\ g}]'\ (\mathsf{un}[\![\mathsf{P}\oplus]\!]\ \{\Gamma\}\ \sigma) \\
\ldots
\end{array}
$$

Note that `reif` is defined by structural induction on `h`. We next wish to generate `h` from Γ. We require

$$\mathsf{allDom} : \forall\,\{\mathsf{p}\}\,(\Gamma : \mathsf{Seq\ p}) \to \mathsf{Dom}\,\Gamma$$

and we create this using a term of type

```
dom : ∀ {p} {A : Fml p} {Γ n} → size (A, Γ) ⩽ n → Dom (A, Γ)
```

Here n is an element of the built-in Agda data-type of natural numbers. We define this term by lexicographical (nested) induction. Agda accepts the outer induction as well-founded because in each call either the Γ or the n argument structurally decreases (the latter only occurs is if $A, \Gamma = \bot, P$ or $\top, N$); and the inner induction because the argument c structurally decreases (the bound on context length).

```
dom : ∀ {p} {A : Fml p} {Γ n} → size (A, Γ) ⩽ n → Dom (A, Γ)
dom {F0} le = DF0
dom {F1} le = DF1
dom {F⊥} (s⩽s (s⩽s le)) = dom⊥ _ _ _ refl le
  where dom⊥ : ∀ c Γ n → c ≡ ctlen Γ → csize (Γ) ⩽ n → Dom (F⊥, Γ)
        -- induction on len Γ (=c)
        dom⊥ zero ε n refl le' = DF⊥
        dom⊥ (suc zero) (_, _ {-} A ε) n' refl le' = D⊥NΓ DF⊥
        dom⊥ (suc zero) (_, _ {+} A ε) n' refl le' = D⊥Pε (dom le')
        dom⊥ zero (A, Γ) n () le'
        dom⊥ (suc zero) ε n' () le'
        ...
        dom⊥ (suc (suc n)) (_, _ {-} M Γ) n' eq le'
          = D⊥NΓ (dom⊥ (suc n) Γ n' (sucinj eq) (⩽lem6 (fsize M) le'))
        dom⊥ (suc (suc n)) (_, _ {+} P (_, _ {-} A Γ)) n' eq le'
          = D⊥PMΓ (dom⊥ (suc n) (P ⊘ A, Γ) n' (sucinj eq) (⩽lem1 (fsize P) le'))
        dom⊥ (suc (suc n)) (_, _ {+} P (_, _ {+} A Γ)) n' eq le'
          = D⊥PQΓ (dom⊥ (suc n) (P par A, Γ) n' (sucinj eq) (⩽lem1 (fsize P) le'))
dom {F⊤} (s⩽s (s⩽s le)) = dom⊤ _ _ _ refl le
  where dom⊤ : ∀ c Γ n → c ≡ ctlen Γ → csize (Γ) ⩽ n → Dom (F⊤, Γ)
        -- induction on len Γ (=c)
        dom⊤ zero ε n refl le' = DF⊤
        dom⊤ (suc zero) (_, _ {+} A ε) n' refl le' = D⊤PΓ DF⊤
        dom⊤ (suc zero) (_, _ {-} A ε) n' refl le' = D⊤Nε (dom le')
        ...
        dom⊤ (suc (suc n)) (_, _ {+} P Γ) n' eq le'
          = D⊤PΓ (dom⊤ (suc n) Γ n' (sucinj eq) (⩽lem6 (fsize P) le'))
        dom⊤ (suc (suc n)) (_, _ {-} N (_, _ {-} A Γ)) n' eq le'
          = D⊤NMΓ (dom⊤ (suc n) (N ⊗ A, Γ) n' (sucinj eq) (⩽lem1 (fsize N) le'))
        dom⊤ (suc (suc n)) (_, _ {-} N (_, _ {+} A Γ)) n' eq le'
```

```
        = DTNQΓ (domT (suc n) (N <| A,Γ) n' (sucinj eq) (⩽lem1 (fsize N) le'))
dom {A = M ⊗ N} le = DM⊗NΓ (dom (⩽lem2 (fsize M) le)) (dom (⩽lem3 (fsize M) le))
dom {A = P par Q} le = DPparQΓ (dom (⩽lem2 (fsize P) le)) (dom (⩽lem3 (fsize P) le))
dom {A = P ⊕ Q} le = DP⊕QΓ (dom (⩽lem4 (fsize P) le)) (dom (⩽lem5 (fsize P) le))
dom {A = M & N} le = DM&NΓ (dom (⩽lem4 (fsize M) le)) (dom (⩽lem5 (fsize M) le))
dom {A = A ⊘ M} le = DA⊘MΓ (dom (⩽lem2 (fsize A) le))
dom {A = A <| P} le = DA<|PΓ (dom (⩽lem2 (fsize A) le))
```

The various lemmas used here are basic facts of natural numbers needed to preserve the bounds, e.g.

```
⩽lem4 : ∀ x {y z n} → suc (suc (x +' y +' z)) ⩽ n → suc (x +' z) ⩽ n
```

We can resultantly define our reification function.

```
allDom : ∀ {p} (Γ : Seq p) → Dom Γ
allDom (A,Γ) = dom (DecTotalOrder.refl decTotalOrder)

reify : ∀ {p} {Γ : Seq p} → Strat p ⟦ Γ ⟧' → ⊢ Γ
reify {p} {Γ} = reif (allDom Γ)

nbe : ∀ {p} {Γ : Seq p} → ⊢ Γ → ⊢ Γ
nbe p = reify ⟦ p ⟧⊢
```

A.2.5 Cut Elimination

We can also formalise the cut elimination procedure in Agda. This is a straightforward implementation of the definitions in Figures 2-12, 2-13, 2-10. We can use Agda as a tool to ensure that all cases are covered and as a type checker. We define the type

```
data ⊢c_ : ∀ {p} → Seq p → Set where
```

representing core proofs on a sequent.

```
mutual
  cut : ∀ {p} {A : Fml p} (Γ : Ctx) {N : Fml -} {P : Fml +} →
    ⊢c A,Γ,,(N ⊥',ε) → ⊢c N,P,ε → ⊢c A,Γ,,(P,ε)
  cut {F1} Γ h g = P1
  cut {F⊥} ε {N} (P⊥+ y) g = P⊥+ $ unpar0 $ cut₂ (N,ε) (wk {P = F0} y) g
  cut {F⊥} (_,_ {-} N Γ') (P⊥- y) g = P⊥- (cut Γ' y g)
  ...
  cut {y & y'} Γ (P& y0 y1) g = P& (cut Γ y0 g) (cut Γ y1 g)
```

```
cut₂ : ∀ (Γ : Seq -) {Q R : Fml +} →
  ⊢c Γ ⊥s, , 0 (Q, ε) → ⊢c Γ, , 0 (R, ε) → ⊢c Q par R, ε
cut₂ (F⊥, ε) (P⊤+ P⊤) (P⊥+ y) = Ppar₂ $ wk y
cut₂ (F⊥, (_, _ {-} N Γ')) (P⊤+ y) (P⊥- y') = cut₂ (_, Γ') y y'
cut₂ (F⊥, _, _ {+} P ε) {Q} {R} (P⊤<| (P⊤- (P<| y))) (P⊥par (P⊥+ (Ppar₁ y')))
 = symPpar $ cut₂ (P ⊥', ε) y'₁ y
  where y'₁ = subst (λ X → ⊢c X, R, ε) (idem⊥F {P, ε}) y'
cut₂ (F⊥, (_, _ {+} P (_, _ {-} M Γ'))) (P⊤<| y) (P⊥⊘ y') = cut₂ (F⊥, P ⊘ M, Γ') y y'
cut₂ (y ⊗ y', y0) (Ppar₁ y1) (P⊗ y2 y3) = cut₂ (y, y', y0) y1 y2
...
```

Termination checking

Unfortunately, Agda cannot be convinced that the above definition passes the termination checker. The problem is not the fact that the induction is not structural (see Section 2.5.1). But rather, the use of subst in the definition of cut_2. To explain, subst has type $A \to (A = B) \to B$, i.e. it takes a term of type A and a proof that $A = B$ and produces a term of type B. This term just returns its first argument, modulo the typing coercions afforded by its second argument. However, to the termination checker of Agda, subst is just any arbitrary function, and so it does not see that $\mathsf{subst}(\sigma, \mathsf{eq})$ is "structurally the same as" σ. This use of subst is essential, and so as far as we can see there is no easy way to get around this problem.

Agda has an experimental feature known as *sized types* [1] which provides support for annotating terms with their *size* (an ordinal) which can be used to solve the above problem. However, we chose not to pursue this here, as it is a non-local solution which involves changing types throughout the development, introducing inelegance.

A.3 A Finitary Programming Language

The above development defines Agda functions between proofs, strategies, and core proofs. Even though we have only formalised the finitary fragment of WS, we can nonetheless embed a version of our call-by-name total programming language inside it. While there is no exponential operator in our formalised fragment of WS, we can define bounded exponentials as in Section 2.2.4. We can resultantly define an imperative programming language in which the use of variables is bounded, and embed this in (our Agda formalisation of) WS. In this language we annotate each variable with a multiplicity, which bounds the number of times it may be used. There are similarities

between this system and [27], which instead places bounds on the number of concurrent threads. We must also restrict to a finitary ground type (Booleans).

A.3.1 Types and Terms

In our finitary language, the type operator $\_ \to \_$ is replaced for an operator $\_ ^n \to \_$ which allows the argument to be called at most n times.

A *context* is a sequence of (variable, type, multiplicity) tuples such that each variable occurs at most once and the multiplicity is a strictly positive number. We say Γ and Δ are *compatible* if they agree on all but the multiplicities, and let $\Gamma + \Delta$ add these multiplicities together. If Γ is a context, let Γ^n be the result of multiplying all multiplicities in Γ by n. An affine lambda calculus based on this notion looks as follows: $\Gamma, x : A^n \vdash s : B$ can be abstracted to $\Gamma \vdash \lambda x.s : A^n \to B$, and we can derive $\Gamma \vdash x : A$ providing $x : A^m \in \Gamma$ for $m \geqslant 1$. For application, we have the following rule:

$$\frac{\Gamma \vdash f : A^n \to B \qquad \Delta \vdash x : A}{\Gamma + \Delta^n \vdash fx : B}$$

The Agda definition of this language is defined below. We omit the definition of compatibility and related lemmas.

```
data FLType : Set where
  Com ♭ool var : FLType
  _£_⇒_ : FLType → ℕ → FLType → FLType

_⊸_ : FLType → FLType → FLType
x ⊸ y = x £ 1 ⇒ y

data FLConst : FLType → Set where
  ifB : FLConst (♭ool ⊸ ♭ool ⊸ ♭ool ⊸ ♭ool)
  ifC : FLConst (♭ool ⊸ Com ⊸ Com ⊸ Com)
  seqB : FLConst (Com ⊸ ♭ool ⊸ ♭ool)
  seqC : FLConst (Com ⊸ Com ⊸ Com)
  cor : ∀ n m → FLConst ((Com £ n ⇒ Com) ⊸ (Com £ m ⇒ Com) ⊸ Com)
  derf : FLConst (var ⊸ ♭ool)
  assgn : FLConst (var ⊸ ♭ool ⊸ Com)
  new : ∀ n A → Bool → FLConst ((var £ n ⇒ A) ⊸ A)
  nott : FLConst (♭ool ⊸ ♭ool)
  repeats : ∀ n → FLConst (Com £ n ⇒ Com)

cmlem : ∀ {a b} → compat a b → (n : ℕ) → compat (mult b n) a
comb : ∀ {Γ Γ'} → compat Γ Γ' → FLCtx
```

```
data FLTerm : FLCtx → FLType → Set where
  konst : ∀ {t Γ} → FLConst t → FLTerm Γ t
  vart : ∀ {t Γ} → (v : Var) → isin v t Γ → FLTerm Γ t
  abs : ∀ {a b Γ n} → (v : Var) → FLTerm ((v , a , n) :: Γ) b → FLTerm Γ (a £ n ⇒ b)
  app : ∀ {a b Γ Δ n} → (c : compat Γ Δ) → FLTerm Γ (a £ n ⇒ b) → FLTerm Δ a
    → FLTerm (comb (cmlem c n)) b
```

Remark In this finitary language, the contravariant bounds can always be inferred from the covariant bounds. More precisely, given a program P with bound annotations only in covariant positions, we can calculate bounds on the contravariant positions such that the program is typeable. This can be shown using a simple induction on terms (for the constants, note that covariant bounds are universally quantified and contravariant bounds given). This corresponds to an *assume-guarantee* lemma in [27].

A.3.2 Language Embedding

We can embed this language inside WS and formalise this embedding in Agda. The key is to use the bounded sequoidal exponentials encountered in Section 2.2.4. We represent the type $A^n \to B$ as $!_n A \multimap B$ where $!_0 A = \mathbf{1}$ and $!_{n+1} A = A \oslash !_n A$.

Bounded Sequoidal Exponentials

We first show how we can model our bounded lambda calculus in WS using the bounded sequoidal exponentials. This involves exhibiting the structure of the linear exponential comonad — modulo the bounds — as proofs in WS.

```
! : ℕ → Fml - → Fml -
! zero M = F1
! (suc n) M = M ⊘ ! n M
  -- Functor
⊗! : ∀ n N → ⊢ ! n N ⊗ N , ! (suc n) N ⊥' , ε
⊗! zero N = P⊗ P1 (Pid⊘ ε P1)
⊗! (suc n) N = P⊗ (P⊘ $ P⊗Tb {Γ = [N] s} $ Pid⊘ ε $ ⊗! n N) (Pid⊘ ε Pid)
prom : ∀ n M N → ⊢ M , N ⊥' , ε → ⊢ ! n M , (! n N) ⊥' , ε
prom zero M N p = P1
prom (suc n) M N p = Pcut {Γ = [M ⊘ ! n M] s} ((N ⊥') <| (! n N ⊥') , ε)
  (PparT {Γ = [M ⊘ ! n M] s} $ Psym {Γ = [M ⊘ ! n M] s} $ P⊘ $
    Pmix {Γ = ε} (N ⊥' , ε) (! n N ⊥' , ε) ε p $
```

```
      P⊗ (P1T {Γ = [! n M] s} $ prom n M N p) P1)
  (⊗! n N)
  -- Dereliction
derel : ∀ A → ⊢ A, ! 1 A ⊥', ε
derel A = P1Tb {Γ = [A] s} $ Pid⊘ ε P1
bang0 : ∀ n → ⊢ ! n F1, F0, ε
bang0 zero = P1
bang0 (suc n) = P⊘ $ P1
  -- Contraction
mutual
  contr₁ : ∀ n m A → ⊢ ! n A, ! m A, (! (n +' m) A) ⊥', ε
  contr₁ zero m A = P1
  contr₁ (suc n) m A = P⊘ $ P⊗Tb {Γ = [A] s} $ Pid⊘ ε $ contr n m A
  contr₂ : ∀ n m A → ⊢ ! m A, ! n A, (! (n +' m) A) ⊥', ε
  contr₂ n m A rewrite +-comm n m = contr₁ m n A
  contr : ∀ n m A → ⊢ ! n A ⊗ ! m A, (! (n +' m) A) ⊥', ε
  contr n m A = P⊗ (contr₁ n m A) (contr₂ n m A)
  -- Multiplication
!mult : ∀ n m A → ⊢ (! n (! m A)), [! (n * m) A ⊥']
!mult zero m A = P1
!mult (suc n) m A = Pcut {Γ = [! m A ⊘ ! n (! m A)] s} (! (m +' (n * m)) A ⊥', ε)
  (P⊘ $ PparT {Γ = ! m A, ! n (! m A), ε} $ Pmix {Γ = ε} (! m A ⊥', ε)
    (! (n * m) A ⊥', ε) ε Pid (P⊗
      (P1T {Γ = [! n (! m A)] s} $ !mult n m A)
        P1))
  (contr m (n * m) A)
  -- Monoidalness
!monoidal : ∀ n M N → ⊢ ! n (M ⊗ N), ! n M ⊥', ! n N ⊥', ε
...
```

Translation of Constants

We next show how each of the program constants can be modelled in WS. We omit coroutine composition, for brevity: as with all omitted details, it can be found at [18].

```
Fcom  =  F⊥ <| F⊤
Fbool  =  F⊥ <| (F⊤ ⊕ F⊤)
Fvar  =  Fbool & (Fcom & Fcom)

  -- Translation of program types to WS formulas

toFml : FLType → Fml -
toFml Com  =  Fcom
toFml bool  =  Fbool
toFml var  =  Fvar
toFml (y £ y' ⇒ y0)  =  (toFml y0) <| ((! y' (toFml y)) ⊥')

  -- Operations on Boolean Expressions

P⊕i : ∀ {A} {Γ} → Bool → ⊢ A, Γ → ⊢ A ⊕ A, Γ
P⊕i true  =  P⊕1
P⊕i false  =  P⊕2

P&i : ∀ {M} {Γ} → (Bool → ⊢ M, Γ) → ⊢ M & M, Γ
P&i f  =  P& (f true) (f false)

constB : Bool → ⊢ Fbool, ε
constB m  =  P<| $ P⊥+ $ P⊕i m $ P⊤

unaryB : (Bool → Bool) → ⊢ Fbool, Fbool ⊥', ε
unaryB f  =  P<| $ P⊥par $ P⊥+ $ Ppar2 $ P⊘ $ P⊤<|
     $ P⊤- $ P<| $ P&i $ λ m → P⊥+ $ P⊕i (f m) $ P⊤

  -- Constants for Imperative Flow

seq : ∀ (P : Fml +) → ⊢ F⊥ <| P, Fcom ⊥', (F⊥ <| P) ⊥', ε
seq P  =  P<| $ P⊥par $ P⊥par $ P⊥+ $ Ppar1 $
  Ppar2 $ P⊘ $ P⊤<| $ P⊤<| $ P⊤- $ P<| $ P<| $
  P⊥par $ P⊥+ $ Ppar2 $ P⊘ $ P⊤<| $ P⊤- $ P<| $
  aux $ sym $ idem⊥f P
  where aux : ∀ {Q} → Q ≡ (P ⊥') ⊥' → ⊢ P ⊥', Q, ε
    aux refl  =  Pid

ifthen : ∀ (P : Fml +) → ⊢ \bot \lhd P, Fbool ⊥', \top \oslash P ⊥', \top \oslash P ⊥', ε
ifthen P  =  P<| $ P⊥par $ P⊥par $ P⊥par $ P⊥+ $ Ppar1
          $ Ppar1 $ Ppar2 $ P⊘ $ P⊤<| $ P⊤<| $ P⊤<|
          $ P⊤- $ P<| $ P<| $ P<| $
            P& (Pstr {Γ = F⊥, P, ε} aux) (Pstr {Γ = F⊥, P, F⊤ ⊘ (P ⊥'), ε} aux)
     where Pid' : ∀ {P : Fml +} → ⊢ P ⊥', P, ε
           Pid' {P}  =  subst (λ X → ⊢ P ⊥', X, ε) (idem⊥f P) (Pid {P ⊥'})
```

```
    aux : ⊢ F⊥, P, F⊤ ⊘ (P ⊥'), ε
    aux = P⊥par $ P⊥+ $ Ppar₂ $ P⊘ $ P⊤<| $ P⊤- $ P<| $ Pid'
repeat : ∀ n → ⊢ Fcom, (! n Fcom) ⊥', ε
repeat n = P<| $ P⊥par $ P⊥+ $ repeat' n
  where repeat' : ∀ n → ⊢ F⊤ par (! n (F⊥ <| F⊤) ⊥'), ε
        repeat' zero = Ppar₁ $ P⊤+ $ P⊤
        repeat' (suc n') = Ppar₂ $ P<| $ P⊘ $ P⊤<| $ P⊤<| $ P⊤- $ P<|
          $ P<| $ Psym {Γ = F⊥, ε} $ P⊥par $ P⊥+ $
          repeat' n'
  -- Finitary Ground Store (of Booleans)
deref : ⊢ Fbool, Fvar ⊥', ε
deref = P⊕T₁ {Γ = [Fbool] s} Pid
assign : ⊢ Fcom, Fbool ⊥', Fvar ⊥', ε
assign = P⊕T₂ {Γ = Fcom, Fbool ⊥', ε} $ P<| $ P⊥par $
        P⊥par $ P⊥+ $ Ppar₁ $ Ppar₂ $ P⊘ $ P⊤<| $ P⊤<| $
        P⊤- $ P<| $ P<| $ P&i $ λ m → P⊥par $ P⊥+ $ Ppar₂ $
        P⊕i m $ P⊘ $ P⊤<| $ P⊤- $ P<| $ P⊥+ $ P⊤
cell : ∀ n → Bool → ⊢ ! n Fvar, ε
cell zero m = P1
cell (suc n) m = P⊘ $ P& (P<| $ P⊥⊘ $ P⊥+ $ P⊘ $ P⊕i m $ P⊤- $ cell n m)
  (P&i $ λ v → P<| $ P⊥⊘ $ P⊥+ $ P⊘ $ P⊤- $ cell n v)
```

Translation of Terms

We can also formalise the lambda-calculus fragment into WS, using the intuitionistic Kleisli embedding as seen in Section 5.4.2, modified to deal with the explicit bounds on the exponentials. This yields an Agda encoding of the map from terms to proofs.

```
toCtx : FLCtx → Ctx
toCtx [] = ε
toCtx ((_, t, n) :: Γ) = (! n (toFml t)) ⊥', toCtx Γ
tToPrf : ∀ {T} {Γ} → FLTerm Γ T → ⊢ toFml T, toCtx Γ
```

Using this we can formalise the game semantics of our language.

```
⟦_⟧σ : ∀ {Γ} {T} → FLTerm Γ T → Strat - ⟦ toFml T, toCtx Γ ⟧'
⟦ t ⟧σ = ⟦ tToPrf t ⟧⊢
```

A.4 Interaction

In this section we describe a tool for running strategies interactively. The idea is that given some strategy σ on a game A, we can ask the machine to "run" the strategy, where the machine plays the role of Player and a human operator the role of Opponent. This corresponds to the "strong evaluation" of [23]. We can use this tool to interact with the game semantics of our finitary programming language.

In Figure A-1 we see an example of this, where the computer is asked to play as Player in the winning strategy representing a three-use Boolean cell. The human playing Opponent first asks to write `False`, then reads from the cell twice.

A.4.1 Annotated Games

Notice that the above demonstration contains more information than that of a strategy on a simple game: in particular, at each node the user is informed of a formula that represents the shape of the given subgame. The interaction module takes as its input not just a strategy σ on a game A, but also an *annotation* on A, of the following type:

```
data Annotation (G : Game) (A : Game → Set) : Set where
  ν : A G → ((i : Mov G) → Annotation (G ≫ i) A) → Annotation G A
```

If A is a constant function, this reduces to

```
data Annotation (G : Game) (A : Set) : Set where
  ν : A → ((i : Mov G) → Annotation (G ≫ i) A) → Annotation G A
```

which annotates each move of G by an element of A. We allow the annotation parameter to be dependent on the game it is annotating: this will allow us to guarantee that when we produce an annotated game from a given formula, the annotation at a particular node is a syntactic representation of that node.

To formalise this, we first need a notion of semantics-annotated syntax. This is a set of formulas parametrised by their (game) semantics.

```
data FML : Pol → Game → Set where
  F0    : FML + I
  F1    : FML - I
  F⊤    : FML + o
  F⊥    : FML - o
  _⊗_   : ∀ {G H} (M : FML - G) (N : FML - H) → FML - (G ⊗' H)
  _par_ : ∀ {G H} (P : FML + G) (Q : FML + H) → FML + (G ⊗' H)
```

Figure A-1: Running Strategies: `Interaction.agda` on a three-use Boolean cell

```
Your choice in '(one + (one + one))'
  encoding (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤)))
    ⊘
    (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤)))
      ⊘
      (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤))) ⊘ F1)))
(options are L0, RL0, RR0) (or "quit") ?
RR0
My choice in '(nil + one)'
  encoding ((F0 par F⊤)
    ⊘
    (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤)))
      ⊘
      (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤))) ⊘ F1)))
  is R0
Your choice in '(nil + (one + (one + one)))'
  encoding ((F1 <| F0)
    ⊗
    (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤)))
      ⊘
      (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤))) ⊘ F1)))
(options are RL0, RRL0, RRR0) (or "quit") ?
RL0
My choice in '(nil + (one + one))'
  encoding (((F0 par (F⊤ ⊕ F⊤))
      ⊘
      (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤))) ⊘ F1))
    ⊘
    (F1 <| F0))
  is RR0
Your choice in '((nil + (one + (one + one))) + nil)'
  encoding (((F1 <| F0)
      ⊗
      (((F⊥ <| (F⊤ ⊕ F⊤)) & ((F⊥ <| F⊤) & (F⊥ <| F⊤))) ⊘ F1))
    ⊗
    (F1 <| F0))
(options are LRL0, LRRL0, LRRR0) (or "quit") ?
LRL0
My choice in '(nil + (one + one))'
  encoding ((((F0 par (F⊤ ⊕ F⊤)) ⊘ F1) ⊘ (F1 <| F0)) ⊘ (F1 <| F0))
  is RR0
You lose, inevitably.
```

```
  _&_ : ∀ {G H} (M : FML - G) (N : FML - H) → FML - (G ×' H)
  _⊕_ : ∀ {G H} (P : FML + G) (Q : FML + H) → FML + (G ×' H)
  _-⊘_ : ∀ {G H} (M : FML - G) (N : FML - H) → FML - (G ⊘' H)
  _+<|_ : ∀ {G H} (P : FML + G) (Q : FML + H) → FML + (G ⊘' H)
  _+⊘_ : ∀ {G H} (P : FML + G) (N : FML - H) → FML + (H ⊸ G)
  _-<|_ : ∀ {G H} (M : FML - G) (P : FML + H) → FML - (H ⊸ G)
PolFML : Game → Set
PolFML G = Σ Pol (λ p → FML p G)
```

We can define simple terms

```
toFml : ∀ {pol} {G} (A : FML pol G) → Fml pol
toFML : ∀ {pol} (A : Fml pol) → FML pol ⟦ A ⟧
```

converting between the annotated version and the purely syntactic version. For this to be well-defined, the annotations given in the definition of FML must coincide with the actual semantic function. We can then define the operation

```
annotate' : ∀ {p} {G} (A : FML p G) → Annotation G PolFML
```

which constructs the semantics of A as an annotated game, with the built-in correctness property defined above. From this we can define

```
forget : ∀ {G} → Annotation G PolFML → Annotation G (λ _ → PolFml)
forget (ν (p, ' A) f) = ν (p , ' toFml A) (λ i → forget (f i))
annotate : ∀ {p} (A : Fml p) → Annotation ⟦ A ⟧ (λ _ → PolFml)
annotate = forget ∘ annotate' ∘ toFML
```

which constructs an annotated game where nodes are annotated by formulas.

A.4.2 Running Strategies

We can define code to "run" a given strategy using the Agda bindings to Haskell and its input-output primitives. The code for this is simple, and is given below (we omit the machinery that deals with forcing the move in the case that the move is unique).

```
mutual
  Ask : ∀ {G A} → Strat - G → Annotation G (λ _ → A) → (A → Doc) → IO Unit
  Ask {gam {mov} f} (neg s) (ν a g) sh with normalise mov | the-mv {mov}
  ... | nil | _ = putDocLn (text "You lose, inevitably.")
```

```
... | one | tm = putDocLn (sep (text "Your choice is forced in"
      :: indent2 (sh a) :: [])) ≫
      Tell (s $ tm refl) (g $ tm refl) sh
... | _ | _  = Ask' {gam {mov} f} (neg s) (ν a g) sh
Ask' : ∀ {G A} → Strat - G → Annotation G (λ _ → A) → (A → Doc) → IO Unit
Ask' {gam {ι} f} (neg s) (ν a g) show =
  prompt (sep (text "Your choice in '" <> prMovEnc ι <> text "'" ::
    indent2 (text "encoding " <> show a) ::
    text "(options are " <> descMoves ι <> text ")" :: []))
    (PC.parseTop (pMov ι))
    (λ i → Tell (s i) (g i) show)
Tell : ∀ {G A} → (Strat +) G → Annotation G (λ _ → A) → (A → Doc) → IO Unit
Tell {gam {ι} f} (pos i s) (ν a g) show =
  putDocLn (sep (text "My choice in '" <> prMovEnc ι <> text "'" ::
    indent2 (text "encoding " <> show a) ::
    indent2 (text "is " <> prMov ι i) :: [])) ≫
  Ask s (g i) show
```

An example interaction is found in Figure A-1.

A.4.3 WS-Tex

We have also included in the code base a function that produces the LaTeX code for a given proof object, which has proved useful throughout this thesis.

A.5 Towards Infinite Games

In this section we discuss how the development above can be lifted to the setting of infinite games, representing (for example) the sequoidal exponential. Little work on this has been done so far: but we here describe some of the basic ideas and difficulties faced.

Remark The development in this section uses Agda's support for coinductive types. At time of writing, this feature is experimental, with all details subject to change. The Agda version used here is 2.2.10.

A.5.1 Infinite Games

An obvious limitation of the Agda encoding is that only finite games can be represented. To recall, our definition of game is as follows:

```
data Game : Set where
  gam : {ι : MovEnc} → (T ι → Game) → Game
```

The semantics of the data keyword in Agda is that initial algebra semantics are used: the set Game consists of all *finite* trees made up out of the data constructor (which can terminate, in this case, if $\mathsf{T}(\iota) = \emptyset$). Resultantly, all elements of Game must be of finite depth.

To consider (possibly) infinite games, we will need to use final coalgebra semantics. The recommended style in Agda for coinductive definitions is to use a special ∞ construct. The resulting definition of Game is as follows:

```
data Game : Set where
  gam : {ι : MovEnc} → (T ι → ∞ Game) → Game
```

Here ∞ is a special Agda type operator used to mark when a recursive argument of a data definition is coinductive. It is equipped with two operations

```
♯_ : ∀ {A : Set} → A → ∞ A
♭ : ∀ {A : Set} → ∞ A → A
```

Here # can be thought of as a delay constructor, and ♭ as a forcing operator. There are limitations in Agda as to when this can be used to ensure termination.

```
mvEnc : Game → MovEnc
mvEnc (gam {a} f) = a

Mov : Game → Set
Mov G = T (mvEnc G)

_»_ : (G : Game) → Mov G → Game
(gam {ι} f) » x = ♭ (f x)

I : Game
I = gam {nil} ⊥-elim

o : Game
o = gam {one} (λ _ → ♯ I)

bangΣ : Game
bangΣ = gam {one} (λ _ → ♯ bangΣ)
```

Note that the final definition passes Agda's termination checker. This is because the recursive call is *guarded* — it occurs under a # delay constructor. We can define the usual binary operations on infinite games.

```
_×'_ : Game → Game → Game
gam {i} f ×' gam {j} g = gam {i + j} ([f,g])
mutual
  _⊸_ : Game → Game → Game
  G ⊸ (gam {i} f) = gam {i} (λ i' → ♯ (♭ (f i') ⊗ G))
  _⊗_ : Game → Game → Game
  (gam {i} f) ⊗ (gam {j} g) = gam {i + j} h
    where h : T (i + j) → ∞ Game
          h (inj₁ x) = ♯ (gam {j} g ⊸ ♭ (f x))
          h (inj₂ y) = ♯ (gam {i} f ⊸ ♭ (g y))
_⊘_ : Game → Game → Game
(gam {i} f) ⊘ G = gam {i} (λ i' → ♯ (G ⊸ (♭ (f i'))))
```

A.5.2 Sequoidal Exponential

We next wish to describe the sequoidal exponential on infinite games. We might wish to write

```
bang : Game → Game
bang G = G ⊘ (bang G)
```

but this does not pass the productivity checker, as the call to bang on the right hand side does not occur under a guard. The reason that the equation above uniquely defines bang G is because $G \oslash \_$ is a guarded operation in the sense that for each m there exists an n such that the first m moves of $!G$ are defined by the n-fold unwrapping of the equation $G \cong G \oslash !G$. Thus, to give the definition of $!G$ we must massage it into such a form so that Agda sees that the inductive call is guarded.

One way to do this is to use the method described in [24] using an embedded language. We define a new object GameP where elements are a mix of syntax and semantics: they may be games, or algebraic operations applied to games. The type GameH represents the elements of GameP that are in "head normal form" i.e. initial set of moves is available, but the rest of the forest may be a GameP rather than a Game.

```
data GameP : Set where
  gam : {ι : MovEnc} → (T ι → ∞ GameP) → GameP
```

```
  _⊸P_ : GameP → GameP → GameP
  _⊗P_ : GameP → GameP → GameP
data GameH : Set where
  gam : {ι : MovEnc} → (T ι → ∞ GameP) → GameH
HtoP : GameH → GameP
HtoP (gam {i} y) = gam {i} y
GtoP : Game → GameP
GtoP (gam {i} y) = gam {i} (λ x → ♯ (GtoP (♭ (y x))))
```

We define a weak-head-normal-form operation which gives the semantics of $\multimap$ and $\otimes$ one step at a time, extracting the initial set of moves from a given (possibly algebraic) element of GameP.

```
whnf : GameP → GameH
whnf (gam {i} y) = gam {i} y
whnf (y ⊸P y') = (whnf y) ⊸' (whnf y')
  where _⊸'_ : GameH → GameH → GameH
    G ⊸' gam {i} y = gam {i} (λ x → ♯ (♭ (y x) ⊗P (HtoP G)))
whnf (y ⊗P y') = (whnf y) ⊗' (whnf y')
  where _⊗'_ : GameH → GameH → GameH
    gam {i} z ⊗' gam {j} y = gam {i + j}
      [(λ zi → ♯ ((gam {j} y) ⊸P (♭ (z zi)))),
        (λ zi → ♯ ((gam {i} z) ⊸P (♭ (y zi))))]
```

We can use these to pass between each of the types Game, GameP and GameH:

```
mutual
  HtoG : GameH → Game
  HtoG (gam {i} y) = gam {i} (λ x → ♯ (PtoG (♭ (y x))))
  PtoG : GameP → Game
  PtoG g = HtoG (whnf g)
```

Finally, this allows us to define the exponential operator in a way that satisfies Agda's productivity checker.

```
bangP : Game → GameP
bangP (gam {i} f) = gam {i} (λ x → ♯ (bangP (gam {i} f) ⊸P GtoP (♭ (f x))))
bang : Game → Game
bang g = PtoG (bangP g)
```

In the definition of bangP, the recursive call is underneath both # and $\multimap$ P. Since $\multimap$ P is a type constructor, the definition passes Agda's productivity checker.

A.5.3 Continued Development

The development of infinite games in this style can be continued. A first step is to define the categorical structure. However, at the time of writing the Agda productivity checker for coinductive definitions is young, and the fact that the recursive call in coinductive definitions cannot occur under arbitrary functions but only type constructors causes problems for readability, as even simple functions such as copairing need to be expanded out. For example, the morphisms pasc and pasc$_\circ$ are defined by mutual induction in our Agda development (see Section A.1.6), and use symmetry of $\otimes$ for brevity. The productivity checker requires that these definitions are expanded, which increases the number of required mutual definitions exponentially. Thus, this seems a good point to postpone further development until support for coinduction in Agda has matured.

A.6 Further Directions

We have seen that these simple forest games are well suited to formalisation in proof assistants. We have sketched how this can be achieved, but there is much that remains to be done here. This includes:

- Continuing the coalgebraic treatment of infinite games
- Considering partial strategies
- Considering first-order structure, as found in Chapter 4
- Formalising proofs of properties of the categories of games — e.g. associativity of strategy composition, sequoidal closedness, ...
- Considering other game models, e.g. Conway games.